KV11 Sub-Family Reference Manual

Supports: MKV11Z128VLH7, MKV11Z128VLF7, MKV11Z128VLC7, MKV11Z128VFM7, MKV11Z64VLH7, MKV11Z64VLF7, MKV11Z64VLC7, MKV11Z64VFM7, MKV10Z64VLH7, MKV10Z64VLF7, MKV10Z64VLC7, MKV10Z64VFM7, MKV10Z128VLH7, MKV10Z128VLF7, MKV10Z128VLC7, MKV10Z128VFM7, MKV11Z128VLH7P, MKV11Z128VLF7P, MKV11Z128VLC7P, MKV11Z128VFM7P, MKV10Z64VLH7P, MKV10Z64VLF7P, MKV10Z64VLC7P, MKV10Z64VFM7P

Document Number: KV11P64M75RM
Rev. 4, May 2017

Contents

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

Chapter 1
About This Document

1.1 Overview....39

1.1.1 Purpose....39

1.1.2 Audience....39

1.2 Conventions....39

1.2.1 Numbering systems....39

1.2.2 Typographic notation....40

1.2.3 Special terms....40

Chapter 2
Introduction

2.1 Overview....41

2.2 Module Functional Categories....41

2.2.1 ARM Cortex-M0+ Core Modules....42

2.2.2 System Modules....43

2.2.3 Memories and Memory Interfaces....44

2.2.4 Clocks....44

2.2.5 Security and Integrity modules....44

2.2.6 Analog modules....44

2.2.7 Timer modules....45

2.2.8 Communication interfaces....46

2.2.9 Human-machine interfaces....46

2.2.10 Kinetis Motor Suite (KMS)....46

2.3 Orderable part numbers....47

Chapter 3
Chip Configuration

3.1 Introduction....49

3.2 Module to Module Interconnects....49

3.2.1 Module to Module interconnections 49
3.3 Core Modules 57
3.3.1 ARM Cortex-M0+ Core Configuration 57
3.3.2 Nested Vectored Interrupt Controller (NVIC) Configuration 60
3.3.3 Asynchronous wake-up interrupt controller (AWIC) configuration 64
3.4 System Modules 65
3.4.1 SIM Configuration 65
3.4.2 Flashloader configuration 66
3.4.3 System Mode Controller (SMC) Configuration 66
3.4.4 PMC Configuration 67
3.4.5 Low-Leakage Wake-up Unit (LLWU) Configuration 68
3.4.6 MMDVSQ Configuration 70
3.4.7 MCM Configuration 71
3.4.8 Crossbar-Light Switch Configuration 72
3.4.9 Peripheral Bridge Configuration 73
3.4.10 DMA request multiplexer configuration 74
3.4.11 DMA Controller Configuration 77
3.4.12 DMA Channel output assignments 77
3.4.13 External Watchdog Monitor (EWM) Configuration 78
3.4.14 Watchdog Configuration 79
3.5 Clock Modules 81
3.5.1 MCG Configuration 81
3.5.2 OSC Configuration 82
3.6 Memories and Memory Interfaces 83
3.6.1 Flash Memory Configuration 83
3.6.2 Flash Memory Controller Configuration 87
3.6.3 SRAM Configuration 88
3.7 Security 90
3.7.1 CRC Configuration 90

3.8 Analog....91
3.8.1 16-bit SAR ADC Configuration....91
3.8.2 CMP Configuration....95
3.8.3 12-bit DAC Configuration....98
3.9 Timers....99
3.9.1 FlexTimer Configuration....99
3.9.2 Low-power timer configuration....108
3.9.3 PDB Configuration....109
3.10 Communication interfaces....113
3.10.1 SPI configuration....113
3.10.2 I2C Configuration....116
3.10.3 UART Configuration....117
3.10.4 FlexCAN Configuration....119
3.11 Human-machine interfaces (HMI)....121
3.11.1 GPIO Configuration....121
3.12 Kinetis Motor Suite Configuration....122
3.12.1 KMS configuration....122
3.12.2 Configuration of the production MCUs....123
3.12.3 KMS Library....124
3.12.4 Library Protection....124
3.12.5 Flash protection....125

Chapter 4
Memory Map

4.1 Introduction....127
4.2 System memory map....127
4.3 Flash Memory Map....128
4.3.1 Alternate Non-Volatile IRC User Trim Description....128
4.4 SRAM memory map....129
4.5 Bit Manipulation Engine....129

4.6 Peripheral bridge (AIPS-Lite) memory map....130
4.6.1 Read-after-write sequence and required serialization of memory operations....130
4.6.2 Peripheral Bridge (AIPS-Lite) Memory Map....131
4.6.3 Modules Restricted Access in User Mode....134
4.7 Private Peripheral Bus (PPB) memory map....134

Chapter 5
Clock Distribution

5.1 Introduction....137
5.2 Programming model....137
5.3 High-level device clocking diagram....137
5.4 Clock definitions....138
5.4.1 Device clock summary....139
5.5 Internal clocking requirements....140
5.5.1 Clock divider values after reset....141
5.5.2 VLPR mode clocking....142
5.6 Clock Gating....142
5.7 Module clocks....142
5.7.1 PMC 1-kHz LPO clock....144
5.7.2 WDOG clocking....144
5.7.3 LPTMR clocking....144
5.7.4 FlexCAN clocking....145
5.7.5 Flex Timer (FTM) clocking....145
5.7.6 UART clocking....146
5.7.7 SPI clocking....147
5.7.8 ADC clocking....148

Chapter 6
Reset and Boot

6.1 Introduction....149
6.2 Reset....149

6.2.1 Power-on reset (POR)..... 150
6.2.2 System reset sources..... 150
6.2.3 MCU Resets..... 153
6.2.4 Reset Pin 154
6.2.5 Debug resets..... 155
6.3 Boot..... 156
6.3.1 Boot sources..... 156
6.3.2 FOPT boot options..... 156
6.3.3 Boot sequence..... 157

Chapter 7
Power Management

7.1 Introduction..... 159
7.2 Clocking Modes..... 159
7.2.1 Partial Stop..... 159
7.2.2 DMA Wakeup..... 160
7.2.3 Compute Operation..... 161
7.2.4 Peripheral Doze..... 162
7.2.5 Clock Gating..... 163
7.3 Power modes..... 163
7.4 Entering and exiting power modes..... 165
7.5 Module Operation in Low Power Modes..... 165

Chapter 8
Security

8.1 Introduction..... 169
8.2 Flash Security..... 169
8.3 Security Interactions with other Modules..... 169
8.3.1 Security Interactions with Debug..... 170

Chapter 9
Debug

9.1 Introduction..... 171

9.2 Debug Port Pin Descriptions....171
9.3 SWD status and control registers....172
9.3.1 MDM-AP Control Register....173
9.3.2 MDM-AP Status Register....174
9.4 Debug Resets....176
9.5 Micro Trace Buffer (MTB)....176
9.6 Debug in Low-Power Modes....177
9.7 Debug & Security....178

Chapter 10
Signal Multiplexing and Signal Descriptions

10.1 Introduction....179
10.2 Signal Multiplexing Integration....179
10.2.1 Port control and interrupt module features....180
10.2.2 Clock gating....181
10.2.3 Signal multiplexing constraints....181
10.3 Pinout....181
10.3.1 KV11 Signal Multiplexing and Pin Assignments....181
10.3.2 KV11 Pinouts....184
10.4 Module Signal Description Tables....188
10.4.1 Core Modules....188
10.4.2 System Modules....189
10.4.3 Clock Modules....189
10.4.4 Memories and Memory Interfaces....189
10.4.5 Analog....189
10.4.6 Timer Modules....191
10.4.7 Communication Interfaces....193
10.4.8 Human-Machine Interfaces (HMI)....194

Chapter 11
Port Control and Interrupts (PORT)

11.1 Introduction....195
11.2 Overview....195
11.2.1 Features....195
11.2.2 Modes of operation....196
11.3 External signal description....197
11.4 Detailed signal description....197
11.5 Memory map and register definition....197
11.5.1 Pin Control Register n (PORT*x*_PCR*n*)....203
11.5.2 Global Pin Control Low Register (PORT*x*_GPCLR)....206
11.5.3 Global Pin Control High Register (PORT*x*_GPCHR)....206
11.5.4 Interrupt Status Flag Register (PORT*x*_ISFR)....207
11.6 Functional description....207
11.6.1 Pin control....207
11.6.2 Global pin control....208
11.6.3 External interrupts....208

Chapter 12
System Integration Module (SIM)

12.1 Introduction....211
12.1.1 Features....211
12.2 Memory map and register definition....212
12.2.1 System Options Register 1 (SIM_SOPT1)....213
12.2.2 System Options Register 2 (SIM_SOPT2)....214
12.2.3 System Options Register 4 (SIM_SOPT4)....215
12.2.4 System Options Register 5 (SIM_SOPT5)....218
12.2.5 Systems Option Register 6 (SIM_SOPT6)....220
12.2.6 System Options Register 7 (SIM_SOPT7)....222
12.2.7 System Options Register 8 (SIM_SOPT8)....225
12.2.8 System Options Register 9 (SIM_SOPT9)....228
12.2.9 System Device Identification Register (SIM_SDID)....229

12.2.10 System Clock Gating Control Register 4 (SIM_SCGC4)....231
12.2.11 System Clock Gating Control Register 5 (SIM_SCGC5)....232
12.2.12 System Clock Gating Control Register 6 (SIM_SCGC6)....234
12.2.13 System Clock Gating Control Register 7 (SIM_SCGC7)....236
12.2.14 System Clock Divider Register 1 (SIM_CLKDIV1)....237
12.2.15 Flash Configuration Register 1 (SIM_FCFG1)....239
12.2.16 Flash Configuration Register 2 (SIM_FCFG2)....240
12.2.17 Unique Identification Register Mid-High (SIM_UIDMH)....241
12.2.18 Unique Identification Register Mid Low (SIM_UIDML)....242
12.2.19 Unique Identification Register Low (SIM_UIDL)....242
12.2.20 WDOG Control Register (SIM_WDOGC)....243
12.3 Functional description....243

Chapter 13
Kinetis Flashloader

13.1 Introduction....245
13.2 Functional Description....246
13.2.1 Memory Maps....246
13.2.2 Start-up Process....247
13.2.3 Clock Configuration....248
13.2.4 Flashloader Protocol....248
13.2.5 Flashloader Packet Types....253
13.2.6 Flashloader Command API....259
13.3 Peripherals Supported....275
13.3.1 I2C Peripheral....275
13.3.2 SPI Peripheral....276
13.3.3 UART Peripheral....278
13.3.4 CAN (or FlexCAN) Peripheral....281
13.4 Get/SetProperty Command Properties....283
13.4.1 Property Definitions....284

13.5 Kinetis Flashloader Status Error Codes..... 286

Chapter 14
System Mode Controller (SMC)

14.1 Introduction..... 289
14.2 Modes of operation..... 289
14.3 Memory map and register descriptions..... 291
14.3.1 Power Mode Protection register (SMC_PMPROT)..... 292
14.3.2 Power Mode Control register (SMC_PMCTRL)..... 293
14.3.3 Stop Control Register (SMC_STOPCTRL)..... 294
14.3.4 Power Mode Status register (SMC_PMSTAT)..... 296
14.4 Functional description..... 296
14.4.1 Power mode transitions..... 296
14.4.2 Power mode entry/exit sequencing..... 298
14.4.3 Run modes..... 300
14.4.4 Wait modes..... 302
14.4.5 Stop modes..... 303
14.4.6 Debug in low power modes..... 305

Chapter 15
Power Management Controller (PMC)

15.1 Introduction..... 307
15.2 Features..... 307
15.3 Low-voltage detect (LVD) system..... 307
15.3.1 LVD reset operation..... 308
15.3.2 LVD interrupt operation..... 308
15.3.3 Low-voltage warning (LVW) interrupt operation..... 308
15.4 I/O retention..... 309
15.5 Memory map and register descriptions..... 309
15.5.1 Low Voltage Detect Status And Control 1 register (PMC_LVDSC1)..... 310
15.5.2 Low Voltage Detect Status And Control 2 register (PMC_LVDSC2)..... 311

15.5.3 Regulator Status And Control register (PMC_REGSC)..... 312

Chapter 16
Low-Leakage Wakeup Unit (LLWU)

16.1 Introduction..... 315
16.1.1 Features..... 315
16.1.2 Modes of operation..... 316
16.1.3 Block diagram..... 316
16.2 LLWU signal descriptions..... 318
16.3 Memory map/register definition..... 318
16.3.1 LLWU Pin Enable 1 register (LLWU_PE1)..... 319
16.3.2 LLWU Pin Enable 2 register (LLWU_PE2)..... 320
16.3.3 LLWU Pin Enable 3 register (LLWU_PE3)..... 321
16.3.4 LLWU Pin Enable 4 register (LLWU_PE4)..... 322
16.3.5 LLWU Pin Enable 5 register (LLWU_PE5)..... 323
16.3.6 LLWU Pin Enable 6 register (LLWU_PE6)..... 325
16.3.7 LLWU Pin Enable 7 register (LLWU_PE7)..... 326
16.3.8 LLWU Pin Enable 8 register (LLWU_PE8)..... 327
16.3.9 LLWU Module Enable register (LLWU_ME)..... 328
16.3.10 LLWU Pin Flag 1 register (LLWU_PF1)..... 329
16.3.11 LLWU Pin Flag 2 register (LLWU_PF2)..... 331
16.3.12 LLWU Pin Flag 3 register (LLWU_PF3)..... 333
16.3.13 LLWU Pin Flag 4 register (LLWU_PF4)..... 334
16.3.14 LLWU Module Flag 5 register (LLWU_MF5)..... 336
16.3.15 LLWU Pin Filter 1 register (LLWU_FILT1)..... 338
16.3.16 LLWU Pin Filter 2 register (LLWU_FILT2)..... 339
16.4 Functional description..... 340
16.4.1 VLLS modes..... 340
16.4.2 Initialization..... 341

Chapter 17

Reset Control Module (RCM)

17.1 Introduction....343
17.2 Reset memory map and register descriptions....343
17.2.1 System Reset Status Register 0 (RCM_SRS0)....344
17.2.2 System Reset Status Register 1 (RCM_SRS1)....345
17.2.3 Reset Pin Filter Control register (RCM_RPFC)....346
17.2.4 Reset Pin Filter Width register (RCM_RPFW)....347

Chapter 18
Bit Manipulation Engine (BME)

18.1 Introduction....349
18.1.1 Overview....350
18.1.2 Features....350
18.1.3 Modes of operation....351
18.2 Memory map and register definition....351
18.3 Functional description....351
18.3.1 BME decorated stores....352
18.3.2 BME decorated loads....359
18.3.3 Additional details on decorated addresses and GPIO accesses....365
18.4 Application information....366

Chapter 19
Memory-Mapped Divide and Square Root (MMDVSQ)

19.1 Introduction....369
19.1.1 Features....369
19.1.2 Block diagram....370
19.1.3 Modes of operation....371
19.2 External signal description....372
19.3 Memory map and register definition....372
19.3.1 Dividend Register (MMDVSQ_DEND)....373
19.3.2 Divisor Register (MMDVSQ_DSOR)....373
19.3.3 Control/Status Register (MMDVSQ_CSR)....375

19.3.4 Result Register (MMDVSQ_RES)..... 378
19.3.5 Radicand Register (MMDVSQ_RCND)..... 378
19.4 Functional description.....379
19.4.1 Algorithms..... 379
19.4.2 Execution times.....382
19.4.3 Software interface..... 384

Chapter 20
Miscellaneous Control Module (MCM)

20.1 Introduction.....387
20.1.1 Features..... 387
20.2 Memory map/register descriptions..... 387
20.2.1 Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC).....388
20.2.2 Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)..... 388
20.2.3 Platform Control Register (MCM_PLACR).....389
20.2.4 Compute Operation Control Register (MCM_CPO)..... 392

Chapter 21
Micro Trace Buffer (MTB)

21.1 Introduction.....395
21.1.1 Overview.....395
21.1.2 Features..... 397
21.1.3 Modes of operation..... 398
21.2 External signal description.....398
21.3 Memory map and register definition.....399
21.3.1 MTB_RAM Memory Map.....400
21.3.2 MTB_DWT Memory Map.....412
21.3.3 System ROM Memory Map.....422

Chapter 22
Crossbar Switch Lite (AXBS-Lite)

22.1 Introduction.....427
22.1.1 Features..... 427

22.2 Memory Map / Register Definition....428
22.3 Functional Description....428
22.3.1 General operation....428
22.3.2 Arbitration....429
22.4 Initialization/application information....430

Chapter 23
Peripheral Bridge (AIPS-Lite)

23.1 Introduction....431
23.1.1 Features....431
23.1.2 General operation....431
23.2 Memory map/register definition....432
23.3 Functional description....432
23.3.1 Access support....432

Chapter 24
Direct Memory Access Multiplexer (DMAMUX)

24.1 Introduction....433
24.1.1 Overview....433
24.1.2 Features....434
24.1.3 Modes of operation....434
24.2 External signal description....435
24.3 Memory map/register definition....435
24.3.1 Channel Configuration register (DMAMUX_CHCFG*n*)....435
24.4 Functional description....436
24.4.1 Always-enabled DMA sources....436
24.5 Initialization/application information....438
24.5.1 Reset....438
24.5.2 Enabling and configuring sources....438

Chapter 25
Enhanced Direct Memory Access (eDMA)

25.1 Introduction....441

25.1.1 eDMA system block diagram ... 441
25.1.2 Block parts ... 442
25.1.3 Features ... 443
25.2 Modes of operation ... 444
25.3 Memory map/register definition ... 445
25.3.1 TCD memory ... 445
25.3.2 TCD initialization ... 445
25.3.3 TCD structure ... 445
25.3.4 Reserved memory and bit fields ... 446
25.3.5 Control Register (DMA_CR) ... 452
25.3.6 Error Status Register (DMA_ES) ... 455
25.3.7 Enable Request Register (DMA_ERQ) ... 457
25.3.8 Enable Error Interrupt Register (DMA_EEI) ... 459
25.3.9 Clear Enable Error Interrupt Register (DMA_CEEI) ... 460
25.3.10 Set Enable Error Interrupt Register (DMA_SEEI) ... 461
25.3.11 Clear Enable Request Register (DMA_CERQ) ... 462
25.3.12 Set Enable Request Register (DMA_SERQ) ... 463
25.3.13 Clear DONE Status Bit Register (DMA_CDNE) ... 464
25.3.14 Set START Bit Register (DMA_SSRT) ... 465
25.3.15 Clear Error Register (DMA_CERR) ... 466
25.3.16 Clear Interrupt Request Register (DMA_CINT) ... 467
25.3.17 Interrupt Request Register (DMA_INT) ... 468
25.3.18 Error Register (DMA_ERR) ... 469
25.3.19 Hardware Request Status Register (DMA_HRS) ... 471
25.3.20 Enable Asynchronous Request in Stop Register (DMA_EARS) ... 473
25.3.21 Channel n Priority Register (DMA_DCHPRI*n*) ... 474
25.3.22 TCD Source Address (DMA_TCD*n*_SADDR) ... 475
25.3.23 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF) ... 475
25.3.24 TCD Transfer Attributes (DMA_TCD*n*_ATTR) ... 476

25.3.25 TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD*n*_NBYTES_MLNO)..... 477
25.3.26 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)..... 477
25.3.27 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)..... 479
25.3.28 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)..... 480
25.3.29 TCD Destination Address (DMA_TCD*n*_DADDR)..... 480
25.3.30 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)..... 481
25.3.31 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)..... 481
25.3.32 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)..... 483
25.3.33 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)..... 484
25.3.34 TCD Control and Status (DMA_TCD*n*_CSR)..... 484
25.3.35 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)..... 487
25.3.36 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)..... 488
25.4 Functional description..... 489
25.4.1 eDMA basic data flow..... 489
25.4.2 Fault reporting and handling..... 492
25.4.3 Channel preemption..... 494
25.4.4 Performance..... 494
25.5 Initialization/application information..... 499
25.5.1 eDMA initialization..... 499
25.5.2 Programming errors..... 501
25.5.3 Arbitration mode considerations..... 501
25.5.4 Performing DMA transfers..... 502
25.5.5 Monitoring transfer descriptor status..... 506
25.5.6 Channel Linking..... 508

25.5.7 Dynamic programming..... 509

Chapter 26
External Watchdog Monitor (EWM)

26.1 Introduction.....513
26.1.1 Features..... 513
26.1.2 Modes of Operation..... 514
26.1.3 Block Diagram..... 515
26.2 EWM Signal Descriptions..... 515
26.3 Memory Map/Register Definition.....516
26.3.1 Control Register (EWM_CTRL)..... 516
26.3.2 Service Register (EWM_SERV).....517
26.3.3 Compare Low Register (EWM_CMPL)..... 517
26.3.4 Compare High Register (EWM_CMPH).....518
26.3.5 Clock Control Register (EWM_CLKCTRL).....519
26.3.6 Clock Prescaler Register (EWM_CLKPRESCALER)..... 519
26.4 Functional Description.....520
26.4.1 The EWM_out Signal..... 520
26.4.2 The EWM_in Signal..... 521
26.4.3 EWM Counter..... 521
26.4.4 EWM Compare Registers..... 522
26.4.5 EWM Refresh Mechanism.....522
26.4.6 EWM Interrupt.....522
26.4.7 Selecting the EWM counter clock..... 523
26.4.8 Counter clock prescaler.....523

Chapter 27
Watchdog Timer (WDOG)

27.1 Introduction.....525
27.2 Features.....525
27.3 Functional overview.....526

27.3.1 Unlocking and updating the watchdog....528
27.3.2 Watchdog configuration time (WCT)....529
27.3.3 Refreshing the watchdog....530
27.3.4 Windowed mode of operation....530
27.3.5 Watchdog disabled mode of operation....530
27.3.6 Low-power modes of operation....530
27.3.7 Debug modes of operation....531
27.4 Testing the watchdog....531
27.4.1 Quick test....532
27.4.2 Byte test....532
27.5 Backup reset generator....534
27.6 Generated resets and interrupts....534
27.7 Memory map and register definition....535
27.7.1 Watchdog Status and Control Register High (WDOG_STCTRLH)....536
27.7.2 Watchdog Status and Control Register Low (WDOG_STCTRLL)....537
27.7.3 Watchdog Time-out Value Register High (WDOG_TOVALH)....538
27.7.4 Watchdog Time-out Value Register Low (WDOG_TOVALL)....538
27.7.5 Watchdog Window Register High (WDOG_WINH)....539
27.7.6 Watchdog Window Register Low (WDOG_WINL)....539
27.7.7 Watchdog Refresh register (WDOG_REFRESH)....540
27.7.8 Watchdog Unlock register (WDOG_UNLOCK)....540
27.7.9 Watchdog Timer Output Register High (WDOG_TMROUTH)....540
27.7.10 Watchdog Timer Output Register Low (WDOG_TMROUTL)....541
27.7.11 Watchdog Reset Count register (WDOG_RSTCNT)....541
27.7.12 Watchdog Prescaler register (WDOG_PRESC)....541
27.8 Watchdog operation with 8-bit access....542
27.8.1 General guideline....542
27.8.2 Refresh and unlock operations with 8-bit access....542
27.9 Restrictions on watchdog operation....543

Chapter 28
Multipurpose Clock Generator (MCG)

28.1 Introduction....547
28.1.1 Features....547
28.1.2 Modes of Operation....549
28.2 External Signal Description....550
28.3 Memory Map/Register Definition....550
28.3.1 MCG Control 1 Register (MCG_C1)....550
28.3.2 MCG Control 2 Register (MCG_C2)....551
28.3.3 MCG Control 3 Register (MCG_C3)....553
28.3.4 MCG Control 4 Register (MCG_C4)....553
28.3.5 MCG Control 5 Register (MCG_C5)....554
28.3.5 MCG Control 6 Register (MCG_C6)....555
28.3.6 MCG Status Register (MCG_S)....555
28.3.7 MCG Status and Control Register (MCG_SC)....556
28.3.8 MCG Auto Trim Compare Value High Register (MCG_ATCVH)....558
28.3.9 MCG Auto Trim Compare Value Low Register (MCG_ATCVL)....558
28.4 Functional description....558
28.4.1 MCG mode state diagram....558
28.4.2 Low-power bit usage....561
28.4.3 MCG Internal Reference Clocks....562
28.4.4 External Reference Clock....562
28.4.5 MCG Fixed Frequency Clock....563
28.4.6 MCG Auto TRIM (ATM)....563
28.5 Initialization / Application information....564
28.5.1 MCG module initialization sequence....565
28.5.2 Using a 32.768 kHz reference....567
28.5.3 MCG mode switching....567

Chapter 29

Oscillator (OSC)

29.1 Introduction....569
29.2 Features and Modes....569
29.3 Block Diagram....570
29.4 OSC Signal Descriptions....570
29.5 External Crystal / Resonator Connections....571
29.6 External Clock Connections....572
29.7 Memory Map/Register Definitions....573
29.7.1 OSC Memory Map/Register Definition....573
29.8 Functional Description....574
29.8.1 OSC module states....574
29.8.2 OSC module modes....576
29.8.3 Counter....578
29.8.4 Reference clock pin requirements....578
29.9 Reset....578
29.10 Low power modes operation....579
29.11 Interrupts....579

Chapter 30
Flash Memory Controller (FMC)

30.1 Introduction....581
30.1.1 Overview....581
30.1.2 Features....581
30.2 Modes of operation....582
30.3 External signal description....582
30.4 Memory map and register descriptions....582
30.5 Functional description....582

Chapter 31
Flash Memory Module (FTFA)

31.1 Introduction....585
31.1.1 Features....586

31.1.2 Block Diagram..... 586
31.1.3 Glossary..... 587
31.2 External Signal Description..... 588
31.3 Memory Map and Registers..... 588
31.3.1 Flash Configuration Field Description..... 588
31.3.2 Program Flash IFR Map..... 589
31.3.3 Register Descriptions..... 590
31.4 Functional Description..... 602
31.4.1 Flash Protection..... 603
31.4.2 Flash Access Protection..... 603
31.4.3 Interrupts..... 605
31.4.4 Flash Operation in Low-Power Modes..... 606
31.4.5 Flash Reads and Ignored Writes..... 606
31.4.6 Read While Write (RWW)..... 606
31.4.7 Flash Program and Erase..... 607
31.4.8 Flash Command Operations..... 607
31.4.9 Margin Read Commands..... 610
31.4.10 Flash Command Description..... 611
31.4.11 Security..... 628
31.4.12 Reset Sequence..... 630

Chapter 32
Cyclic Redundancy Check (CRC)

32.1 Introduction..... 631
32.1.1 Features..... 631
32.1.2 Block diagram..... 631
32.1.3 Modes of operation..... 632
32.2 Memory map and register descriptions..... 632
32.2.1 CRC Data register (CRC_DATA)..... 633
32.2.2 CRC Polynomial register (CRC_GPOLY)..... 634

32.2.3 CRC Control register (CRC_CTRL)....634
32.3 Functional description....635
32.3.1 CRC initialization/reinitialization....635
32.3.2 CRC calculations....636
32.3.3 Transpose feature....637
32.3.4 CRC result complement....639

Chapter 33
Analog-to-Digital Converter (ADC)

33.1 Introduction....641
33.1.1 Features....641
33.1.2 Block diagram....642
33.2 ADC signal descriptions....643
33.2.1 Analog Power (VDDA)....644
33.2.2 Analog Ground (VSSA)....644
33.2.3 Voltage Reference Select....644
33.2.4 Analog Channel Inputs (ADx)....645
33.2.5 Differential Analog Channel Inputs (DADx)....645
33.3 Memory map and register definitions....645
33.3.1 ADC Status and Control Registers 1 (ADC*x*_SC1*n*)....647
33.3.2 ADC Configuration Register 1 (ADC*x*_CFG1)....651
33.3.3 ADC Configuration Register 2 (ADC*x*_CFG2)....653
33.3.4 ADC Data Result Register (ADC*x*_R*n*)....654
33.3.5 Compare Value Registers (ADC*x*_CV*n*)....655
33.3.6 Status and Control Register 2 (ADC*x*_SC2)....656
33.3.7 Status and Control Register 3 (ADC*x*_SC3)....658
33.3.8 ADC Offset Correction Register (ADC*x*_OFS)....659
33.3.9 ADC Plus-Side Gain Register (ADC*x*_PG)....660
33.3.10 ADC Minus-Side Gain Register (ADC*x*_MG)....661
33.3.11 ADC Plus-Side General Calibration Value Register (ADC*x*_CLPD)....661

33.3.12 ADC Plus-Side General Calibration Value Register (ADC*x*_CLPS)..... 662
33.3.13 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP4)..... 662
33.3.14 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP3)..... 663
33.3.15 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP2)..... 663
33.3.16 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP1)..... 664
33.3.17 ADC Plus-Side General Calibration Value Register (ADC*x*_CLP0)..... 664
33.3.18 ADC Minus-Side General Calibration Value Register (ADC*x*_CLMD)..... 665
33.3.19 ADC Minus-Side General Calibration Value Register (ADC*x*_CLMS)..... 665
33.3.20 ADC Minus-Side General Calibration Value Register (ADC*x*_CLM4)..... 666
33.3.21 ADC Minus-Side General Calibration Value Register (ADC*x*_CLM3)..... 666
33.3.22 ADC Minus-Side General Calibration Value Register (ADC*x*_CLM2)..... 667
33.3.23 ADC Minus-Side General Calibration Value Register (ADC*x*_CLM1)..... 667
33.3.24 ADC Minus-Side General Calibration Value Register (ADC*x*_CLM0)..... 668
33.4 Functional description..... 668
33.4.1 Clock select and divide control..... 669
33.4.2 Voltage reference selection..... 670
33.4.3 Hardware trigger and channel selects..... 670
33.4.4 Conversion control..... 671
33.4.5 Automatic compare function..... 679
33.4.6 Calibration function..... 680
33.4.7 User-defined offset function..... 682
33.4.8 Temperature sensor..... 683
33.4.9 MCU wait mode operation..... 684
33.4.10 MCU Normal Stop mode operation..... 684
33.4.11 MCU Low-Power Stop mode operation..... 685
33.5 Initialization information..... 686
33.5.1 ADC module initialization example..... 686
33.6 Application information..... 688
33.6.1 External pins and routing..... 688

33.6.2 Sources of error....690

Chapter 34
Comparator (CMP)

34.1 Introduction....695
34.1.1 CMP features....695
34.1.2 6-bit DAC key features....696
34.1.3 ANMUX key features....696
34.1.4 CMP, DAC and ANMUX diagram....697
34.1.5 CMP block diagram....698
34.2 Memory map/register definitions....700
34.2.1 CMP Control Register 0 (CMP*x*_CR0)....700
34.2.2 CMP Control Register 1 (CMP*x*_CR1)....701
34.2.3 CMP Filter Period Register (CMP*x*_FPR)....703
34.2.4 CMP Status and Control Register (CMP*x*_SCR)....703
34.2.5 DAC Control Register (CMP*x*_DACCR)....704
34.2.6 MUX Control Register (CMP*x*_MUXCR)....705
34.3 Functional description....706
34.3.1 CMP functional modes....706
34.3.2 Power modes....715
34.3.3 Startup and operation....716
34.3.4 Low-pass filter....716
34.4 CMP interrupts....719
34.5 DMA support....719
34.6 CMP Asynchronous DMA support....719
34.7 Digital-to-analog converter....720
34.8 DAC functional description....720
34.8.1 Voltage reference source select....720
34.9 DAC resets....721
34.10 DAC clocks....721

34.11 DAC interrupts....721
34.12 CMP Trigger Mode....721

Chapter 35
12-bit Digital-to-Analog Converter (DAC)

35.1 Introduction....723
35.2 Features....723
35.3 Block diagram....723
35.4 Memory map/register definition....724
35.4.1 DAC Data Low Register (DAC*x*_DAT*n*L)....725
35.4.2 DAC Data High Register (DAC*x*_DAT*n*H)....725
35.4.3 DAC Status Register (DAC*x*_SR)....726
35.4.4 DAC Control Register (DAC*x*_C0)....727
35.4.5 DAC Control Register 1 (DAC*x*_C1)....728
35.4.6 DAC Control Register 2 (DAC*x*_C2)....729
35.5 Functional description....729
35.5.1 DAC data buffer operation....729
35.5.2 DMA operation....731
35.5.3 Resets....731
35.5.4 Low-Power mode operation....731

Chapter 36
FlexTimer Module (FTM)

36.1 Introduction....733
36.1.1 FlexTimer philosophy....733
36.1.2 Features....734
36.1.3 Modes of operation....735
36.1.4 Block diagram....736
36.2 FTM signal descriptions....738
36.3 Memory map and register definition....738
36.3.1 Memory map....738

36.3.2 Register descriptions....739
36.3.3 Status And Control (FTMx_SC)....748
36.3.4 Counter (FTMx_CNT)....749
36.3.5 Modulo (FTMx_MOD)....750
36.3.6 Channel (n) Status And Control (FTMx_CnSC)....751
36.3.7 Channel (n) Value (FTMx_CnV)....754
36.3.8 Counter Initial Value (FTMx_CNTIN)....754
36.3.9 Capture And Compare Status (FTMx_STATUS)....755
36.3.10 Features Mode Selection (FTMx_MODE)....757
36.3.11 Synchronization (FTMx_SYNC)....759
36.3.12 Initial State For Channels Output (FTMx_OUTINIT)....761
36.3.13 Output Mask (FTMx_OUTMASK)....762
36.3.14 Function For Linked Channels (FTMx_COMBINE)....764
36.3.15 Deadtime Insertion Control (FTMx_DEADTIME)....769
36.3.16 FTM External Trigger (FTMx_EXTTRIG)....770
36.3.17 Channels Polarity (FTMx_POL)....772
36.3.18 Fault Mode Status (FTMx_FMS)....774
36.3.19 Input Capture Filter Control (FTMx_FILTER)....776
36.3.20 Fault Control (FTMx_FLTCTRL)....777
36.3.21 Quadrature Decoder Control And Status (FTMx_QDCTRL)....780
36.3.22 Configuration (FTMx_CONF)....782
36.3.23 FTM Fault Input Polarity (FTMx_FLTPOL)....783
36.3.24 Synchronization Configuration (FTMx_SYNCONF)....784
36.3.25 FTM Inverting Control (FTMx_INVCTRL)....786
36.3.26 FTM Software Output Control (FTMx_SWOCTRL)....787
36.3.27 FTM PWM Load (FTMx_PWMLOAD)....790
36.4 Functional description....791
36.4.1 Clock source....792
36.4.2 Prescaler....793

36.4.3 Counter....793
36.4.4 Input Capture mode....799
36.4.5 Output Compare mode....803
36.4.6 Edge-Aligned PWM (EPWM) mode....804
36.4.7 Center-Aligned PWM (CPWM) mode....806
36.4.8 Combine mode....808
36.4.9 Complementary mode....815
36.4.10 Registers updated from write buffers....816
36.4.11 PWM synchronization....818
36.4.12 Inverting....834
36.4.13 Software output control....835
36.4.14 Deadtime insertion....837
36.4.15 Output mask....840
36.4.16 Fault control....840
36.4.17 Polarity control....844
36.4.18 Initialization....845
36.4.19 Features priority....845
36.4.20 Channel trigger output....846
36.4.21 Initialization trigger....847
36.4.22 Capture Test mode....850
36.4.23 DMA....850
36.4.24 Dual Edge Capture mode....851
36.4.25 Quadrature Decoder mode....859
36.4.26 BDM mode....864
36.4.27 Intermediate load....865
36.4.28 Global time base (GTB)....867
36.5 Reset overview....869
36.6 FTM Interrupts....870
36.6.1 Timer Overflow Interrupt....871

36.6.2 Channel (n) Interrupt....871
36.6.3 Fault Interrupt....871
36.7 Initialization Procedure....871

Chapter 37
Low-Power Timer (LPTMR)

37.1 Introduction....873
37.1.1 Features....873
37.1.2 Modes of operation....873
37.2 LPTMR signal descriptions....874
37.2.1 Detailed signal descriptions....874
37.3 Memory map and register definition....874
37.3.1 Low Power Timer Control Status Register (LPTMR*x*_CSR)....875
37.3.2 Low Power Timer Prescale Register (LPTMR*x*_PSR)....876
37.3.3 Low Power Timer Compare Register (LPTMR*x*_CMR)....878
37.3.4 Low Power Timer Counter Register (LPTMR*x*_CNR)....878
37.4 Functional description....879
37.4.1 LPTMR power and reset....879
37.4.2 LPTMR clocking....879
37.4.3 LPTMR prescaler/glitch filter....879
37.4.4 LPTMR compare....881
37.4.5 LPTMR counter....881
37.4.6 LPTMR hardware trigger....882
37.4.7 LPTMR interrupt....882

Chapter 38
Programmable Delay Block (PDB)

38.1 Introduction....883
38.1.1 Features....883
38.1.2 Implementation....884
38.1.3 Back-to-back acknowledgment connections....885

38.1.4 DAC External Trigger Input Connections ... 885
38.1.5 Block diagram ... 885
38.1.6 Modes of operation ... 887
38.2 PDB signal descriptions ... 887
38.3 Memory map and register definition ... 887
38.3.1 Status and Control register (PDB*x*_SC) ... 889
38.3.2 Modulus register (PDB*x*_MOD) ... 892
38.3.3 Counter register (PDB*x*_CNT) ... 892
38.3.4 Interrupt Delay register (PDB*x*_IDLY) ... 893
38.3.5 Channel n Control register 1 (PDB*x*_CH*n*C1) ... 893
38.3.6 Channel n Status register (PDB*x*_CH*n*S) ... 894
38.3.7 Channel n Delay 0 register (PDB*x*_CH*n*DLY0) ... 895
38.3.8 Channel n Delay 1 register (PDB*x*_CH*n*DLY1) ... 896
38.3.9 DAC Interval Trigger n Control register (PDB*x*_DACINTC*n*) ... 896
38.3.10 DAC Interval n register (PDB*x*_DACINT*n*) ... 897
38.3.11 Pulse-Out n Enable register (PDB*x*_POEN) ... 898
38.3.12 Pulse-Out n Delay register (PDB*x*_PO*n*DLY) ... 898
38.4 Functional description ... 899
38.4.1 PDB pre-trigger and trigger outputs ... 899
38.4.2 PDB trigger input source selection ... 901
38.4.3 DAC interval trigger outputs ... 901
38.4.4 Pulse-Out's ... 902
38.4.5 Updating the delay registers ... 903
38.4.6 Interrupts ... 905
38.4.7 DMA ... 905
38.5 Application information ... 905
38.5.1 Impact of using the prescaler and multiplication factor on timing resolution ... 905

Chapter 39
CAN (FlexCAN)

Section number	Title	Page

39.1 Introduction....907

39.1.1 Overview....908

39.1.2 FlexCAN module features....909

39.1.3 Modes of operation....910

39.2 FlexCAN signal descriptions....912

39.2.1 CAN Rx....912

39.2.2 CAN Tx....913

39.3 Memory map/register definition....913

39.3.1 FlexCAN memory mapping....913

39.3.2 Module Configuration Register (CAN*x*_MCR)....916

39.3.3 Control 1 register (CAN*x*_CTRL1)....921

39.3.4 Free Running Timer (CAN*x*_TIMER)....925

39.3.5 Rx Mailboxes Global Mask Register (CAN*x*_RXMGMASK)....926

39.3.6 Rx 14 Mask register (CAN*x*_RX14MASK)....927

39.3.7 Rx 15 Mask register (CAN*x*_RX15MASK)....928

39.3.8 Error Counter (CAN*x*_ECR)....928

39.3.9 Error and Status 1 register (CAN*x*_ESR1)....930

39.3.10 Interrupt Masks 1 register (CAN*x*_IMASK1)....936

39.3.11 Interrupt Flags 1 register (CAN*x*_IFLAG1)....936

39.3.12 Control 2 register (CAN*x*_CTRL2)....939

39.3.13 Error and Status 2 register (CAN*x*_ESR2)....942

39.3.14 CRC Register (CAN*x*_CRCR)....944

39.3.15 Rx FIFO Global Mask register (CAN*x*_RXFGMASK)....944

39.3.16 Rx FIFO Information Register (CAN*x*_RXFIR)....945

39.3.17 CAN Bit Timing Register (CAN*x*_CBT)....946

39.3.18 Rx Individual Mask Registers (CAN*x*_RXIMR*n*)....948

39.3.36 Message buffer structure....949

39.3.37 Rx FIFO structure....954

39.4 Functional description....957

39.4.1 Transmit process ... 957
39.4.2 Arbitration process ... 958
39.4.3 Receive process ... 962
39.4.4 Matching process ... 964
39.4.5 Move process ... 969
39.4.6 Data coherence ... 970
39.4.7 Rx FIFO ... 974
39.4.8 CAN protocol related features ... 977
39.4.9 Clock domains and restrictions ... 985
39.4.10 Modes of operation details ... 986
39.4.11 Interrupts ... 991
39.4.12 Bus interface ... 992
39.5 Initialization/application information ... 993
39.5.1 FlexCAN initialization sequence ... 993

Chapter 40
Serial Peripheral Interface (SPI)

40.1 Introduction ... 995
40.1.1 Block Diagram ... 995
40.1.2 Features ... 996
40.1.3 Interface configurations ... 998
40.1.4 Modes of Operation ... 998
40.2 Module signal descriptions ... 1000
40.2.1 PCS0/SS—Peripheral Chip Select/Slave Select ... 1000
40.2.2 PCS1–PCS3—Peripheral Chip Selects 1–3 ... 1001
40.2.3 SCK—Serial Clock ... 1001
40.2.4 SIN—Serial Input ... 1001
40.2.5 SOUT—Serial Output ... 1001
40.3 Memory Map/Register Definition ... 1001
40.3.1 Module Configuration Register (SPI*x*_MCR) ... 1003

40.3.2 Transfer Count Register (SPI*x*_TCR)..... 1007
40.3.3 Clock and Transfer Attributes Register (In Master Mode) (SPI*x*_CTAR*n*)..... 1007
40.3.4 Clock and Transfer Attributes Register (In Slave Mode) (SPI*x*_CTAR*n*_SLAVE)..... 1012
40.3.5 Status Register (SPI*x*_SR)..... 1013
40.3.6 DMA/Interrupt Request Select and Enable Register (SPI*x*_RSER)..... 1016
40.3.7 PUSH TX FIFO Register In Master Mode (SPI*x*_PUSHR)..... 1018
40.3.8 PUSH TX FIFO Register In Slave Mode (SPI*x*_PUSHR_SLAVE)..... 1020
40.3.9 POP RX FIFO Register (SPI*x*_POPR)..... 1020
40.3.10 Transmit FIFO Registers (SPI*x*_TXFR*n*)..... 1021
40.3.11 Receive FIFO Registers (SPI*x*_RXFR*n*)..... 1021
40.4 Functional description..... 1022
40.4.1 Start and Stop of module transfers..... 1023
40.4.2 Serial Peripheral Interface (SPI) configuration..... 1023
40.4.3 Module baud rate and clock delay generation..... 1028
40.4.4 Transfer formats..... 1030
40.4.5 Continuous Serial Communications Clock..... 1039
40.4.6 Slave Mode Operation Constraints..... 1041
40.4.7 Interrupts/DMA requests..... 1041
40.4.8 Power saving features..... 1044
40.5 Initialization/application information..... 1045
40.5.1 How to manage queues..... 1045
40.5.2 Switching Master and Slave mode..... 1046
40.5.3 Initializing Module in Master/Slave Modes..... 1046
40.5.4 Baud rate settings..... 1046
40.5.5 Delay settings..... 1047
40.5.6 Calculation of FIFO pointer addresses..... 1048

Chapter 41
Inter-Integrated Circuit (I2C)

41.1 Introduction..... 1051

41.1.1 Features ... 1051
41.1.2 Modes of operation ... 1052
41.1.3 Block diagram ... 1052
41.2 I2C signal descriptions ... 1053
41.3 Memory map/register definition ... 1054
41.3.1 I2C Address Register 1 (I2C*x*_A1) ... 1054
41.3.2 I2C Frequency Divider register (I2C*x*_F) ... 1055
41.3.3 I2C Control Register 1 (I2C*x*_C1) ... 1056
41.3.4 I2C Status register (I2C*x*_S) ... 1057
41.3.5 I2C Data I/O register (I2C*x*_D) ... 1059
41.3.6 I2C Control Register 2 (I2C*x*_C2) ... 1060
41.3.7 I2C Programmable Input Glitch Filter Register (I2C*x*_FLT) ... 1061
41.3.8 I2C Range Address register (I2C*x*_RA) ... 1062
41.3.9 I2C SMBus Control and Status register (I2C*x*_SMB) ... 1063
41.3.10 I2C Address Register 2 (I2C*x*_A2) ... 1065
41.3.11 I2C SCL Low Timeout Register High (I2C*x*_SLTH) ... 1065
41.3.12 I2C SCL Low Timeout Register Low (I2C*x*_SLTL) ... 1065
41.4 Functional description ... 1066
41.4.1 I2C protocol ... 1066
41.4.2 10-bit address ... 1071
41.4.3 Address matching ... 1073
41.4.4 System management bus specification ... 1074
41.4.5 Resets ... 1076
41.4.6 Interrupts ... 1076
41.4.7 Programmable input glitch filter ... 1079
41.4.8 Address matching wake-up ... 1079
41.4.9 DMA support ... 1080
41.5 Initialization/application information ... 1080

Chapter 42

Universal Asynchronous Receiver/Transmitter (UART)

42.1 Introduction....1085

42.1.1 Features....1085

42.1.2 Modes of operation....1087

42.2 UART signal descriptions....1088

42.2.1 Detailed signal descriptions....1088

42.3 Memory map and registers....1089

42.3.1 UART Baud Rate Registers: High (UART*x*_BDH)....1091

42.3.2 UART Baud Rate Registers: Low (UART*x*_BDL)....1092

42.3.3 UART Control Register 1 (UART*x*_C1)....1092

42.3.4 UART Control Register 2 (UART*x*_C2)....1094

42.3.5 UART Status Register 1 (UART*x*_S1)....1096

42.3.6 UART Status Register 2 (UART*x*_S2)....1099

42.3.7 UART Control Register 3 (UART*x*_C3)....1100

42.3.8 UART Data Register (UART*x*_D)....1102

42.3.9 UART Match Address Registers 1 (UART*x*_MA1)....1103

42.3.10 UART Match Address Registers 2 (UART*x*_MA2)....1103

42.3.11 UART Control Register 4 (UART*x*_C4)....1103

42.3.12 UART Control Register 5 (UART*x*_C5)....1104

42.3.13 UART Extended Data Register (UART*x*_ED)....1105

42.3.14 UART Modem Register (UART*x*_MODEM)....1106

42.3.15 UART FIFO Parameters (UART*x*_PFIFO)....1107

42.3.16 UART FIFO Control Register (UART*x*_CFIFO)....1109

42.3.17 UART FIFO Status Register (UART*x*_SFIFO)....1110

42.3.18 UART FIFO Transmit Watermark (UART*x*_TWFIFO)....1111

42.3.19 UART FIFO Transmit Count (UART*x*_TCFIFO)....1112

42.3.20 UART FIFO Receive Watermark (UART*x*_RWFIFO)....1112

42.3.21 UART FIFO Receive Count (UART*x*_RCFIFO)....1113

42.4 Functional description....1113

42.4.1 Transmitter....1113
42.4.2 Receiver....1119
42.4.3 Baud rate generation....1132
42.4.4 Data format....1134
42.4.5 Single-wire operation....1137
42.4.6 Loop operation....1138
42.5 Reset....1138
42.6 System level interrupt sources....1138
42.6.1 RXEDGIF description....1139
42.7 DMA operation....1140
42.8 Application information....1140
42.8.1 Transmit/receive data buffer operation....1140
42.8.2 Initialization sequence....1141
42.8.3 Overrun (OR) flag implications....1142
42.8.4 Match address registers....1143
42.8.5 Modem feature....1143
42.8.6 Legacy and reverse compatibility considerations....1145

Chapter 43
General-Purpose Input/Output (GPIO)

43.1 Introduction....1147
43.1.1 Features....1147
43.1.2 Modes of operation....1147
43.1.3 GPIO signal descriptions....1148
43.2 Memory map and register definition....1149
43.2.1 Port Data Output Register (GPIO*x*_PDOR)....1151
43.2.2 Port Set Output Register (GPIO*x*_PSOR)....1152
43.2.3 Port Clear Output Register (GPIO*x*_PCOR)....1152
43.2.4 Port Toggle Output Register (GPIO*x*_PTOR)....1153
43.2.5 Port Data Input Register (GPIO*x*_PDIR)....1153

43.2.6 Port Data Direction Register (GPIO*x*_PDDR)....1154
43.3 FGPIO memory map and register definition....1154
43.3.1 Port Data Output Register (FGPIO*x*_PDOR)....1156
43.3.2 Port Set Output Register (FGPIO*x*_PSOR)....1157
43.3.3 Port Clear Output Register (FGPIO*x*_PCOR)....1157
43.3.4 Port Toggle Output Register (FGPIO*x*_PTOR)....1158
43.3.5 Port Data Input Register (FGPIO*x*_PDIR)....1158
43.3.6 Port Data Direction Register (FGPIO*x*_PDDR)....1159
43.4 Functional description....1159
43.4.1 General-purpose input....1159
43.4.2 General-purpose output....1159
43.4.3 IOPORT....1160

Chapter 44
Release Notes for Revision 4

44.1 General changes throughout document....1161
44.2 About This Document chapter changes....1161
44.3 Introduction chapter changes....1161
44.4 Chip Configuration chapter changes....1161
44.5 Memory Map chapter changes....1162
44.6 Clock Distribution chapter changes....1162
44.7 Reset and Boot chapter changes....1162
44.8 Power Management chapter changes....1162
44.9 Security chapter changes....1162
44.10 Debug chapter changes....1162
44.11 Signal Multiplexing and Signal Descriptions chapter changes....1162
44.12 Port Control and Interrupts (PORT) changes....1163
44.13 SIM changes....1163
44.14 Kinetis Flash Bootloader changes....1163
44.15 System Mode Controller changes (SMC)....1163

44.16 PMC changes ... 1163
44.17 LLWU changes ... 1163
44.18 Reset Control Module changes (RCM) ... 1164
44.19 BME configuration changes ... 1164
44.20 MMDVSQ changes ... 1164
44.21 MCM changes ... 1164
44.22 MTB configuration changes ... 1164
44.23 Crossbar switch module changes ... 1164
44.24 AIPS-Lite module changes ... 1165
44.25 DMAMUX module changes ... 1165
44.26 eDMA module changes ... 1165
44.27 EWM changes ... 1165
44.28 WDOG changes ... 1165
44.29 MCG changes ... 1166
44.30 OSC changes ... 1166
44.31 FMC changes ... 1166
44.32 FTFA changes ... 1166
44.33 CRC changes ... 1166
44.34 ADC changes ... 1167
44.35 CMP changes ... 1167
44.36 DAC changes ... 1167
44.37 FTM changes ... 1167
44.38 LPTMR changes ... 1168
44.39 PDB changes ... 1168
44.40 SPI module changes ... 1169
44.41 I2C changes ... 1169
44.42 FlexCAN module changes ... 1169
44.43 UART changes ... 1170
44.44 GPIO changes ... 1170

Chapter 1
About This Document

1.1 Overview

1.1.1 Purpose

This document describes the features, architecture, and programming model of the NXP KV11 microcontroller.

1.1.2 Audience

This document is primarily for system architects and software application developers who are using or considering using the microcontroller 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.
w1c	Write 1 to clear: Refers to a register bitfield that must be written as 1 to be "cleared."

Chapter 2
Introduction

2.1 Overview

This chapter provides high-level descriptions of the modules available on the devices covered by this document.

2.2 Module Functional Categories

The modules on this device are grouped into functional categories. The following sections describe the modules assigned to each category in more detail.

Table 2-1. Module functional categories

Module category	Description

ARM® Cortex®-M0+ core	• 32-bit MCU core from ARM's Cortex-M class, 1.77 CoreMark®/MHz from single-cycle access memories, 75 MHz CPU frequency
	• Additional 32-bit integer divide and square root arithmetic module
	  • Supports 32/32 signed and unsigned divide (or remainder) calculations
	  • Supports 32-bit unsigned square root calculations

System	• System integration module • Power management and mode controllers • Multiple power modes available based on run, wait, stop, and power-down modes • Low-leakage wakeup unit • Miscellaneous control module • Crossbar switch • Peripheral bridge • Direct memory access (DMA) controller with multiplexer to increase available DMA requests • External watchdog monitor • Watchdog • Kinetis Flashloader

Memories	• Internal memories include:
	  • Up to 128/64 KB flash memory
	  • 16 KB SRAM

Table continues on the next page...

Table 2-1. Module functional categories (continued)

Module category	Description
Clocks	• Multiple clock generation options available from internally- and externally-generated clocks • System oscillator to provide clock source for the MCU • MCG module with FLL for systems and CPU clock sources
Security	• Cyclic Redundancy Check module for error detection
Analog	• 16-bit analog-to-digital converter • Comparator • Digital-to-analog converter
Timers	• Programmable delay block • FlexTimers • Low power timer
Communications	• Serial peripheral interface • Inter-integrated circuit (I^2C) • UART • FlexCAN
Human-Machine Interfaces (HMI)	• General purpose input/output controller

2.2.1 ARM Cortex-M0+ Core Modules

The following core modules are available on this device.

Table 2-2. Core modules

Module	Description
ARM Cortex-M0+	The ARM® Cortex™-M0+ is the newest member of the Cortex M Series of processors targeting microcontroller applications focused on very cost sensitive, deterministic, interrupt driven environments. The Cortex M0+ processor is based on the ARMv6 Architecture and Thumb®-2 ISA and is 100% instruction set compatible with its predecessor, the Cortex-M0 core, and upward compatible to Cortex-M3 and M4 cores.
NVIC	The ARMv6-M exception model and nested-vectored interrupt controller (NVIC) implement a relocatable vector table supporting many external interrupts, a single non-maskable interrupt (NMI), and priority levels. The NVIC replaces shadow registers with equivalent system and simplified programmability. The NVIC contains the address of the function to execute for a particular handler. The address is fetched via the instruction port allowing parallel register stacking and look-up. The first sixteen entries are allocated to ARM internal sources with the others mapping to MCU-defined interrupts.
AWIC	The primary function of the Asynchronous Wake-up Interrupt Controller (AWIC) is to detect asynchronous wake-up events in stop modes and signal to clock control logic to resume system clocking. After clock restart, the NVIC observes the pending interrupt and performs the normal interrupt or event processing.
Single-cycle I/O Port	For high-speed, single-cycle access to peripherals, the Cortex-M0+ processor implements a dedicated single-cycle I/O port.
Debug interfaces	Most of this device's debug is based on the ARM CoreSight™ architecture. One debug interfaces are supported:

Table continues on the next page...

Table 2-2. Core modules (continued)

Module	Description
	• Serial Wire Debug (SWD) • Micro Trace Buffer (MTB)
MMDVSQ	32-bit integer divide and square root arithmetic module • Supports 32/32 signed and unsigned divide (or remainder) calculations • Supports 32-bit unsigned square root calculations

2.2.2 System Modules

The following system modules are available on this device.

Table 2-3. System modules

Module	Description
System integration module (SIM)	The SIM includes integration logic and several module configuration settings.
System mode controller	The SMC provides control and protection on entry and exit to each power mode, control for the Power management controller (PMC), and reset entry and exit for the complete MCU.
Power management controller (PMC)	The PMC provides the user with multiple power options. Ten different modes are supported that allow the user to optimize power consumption for the level of functionality needed. Includes power-on-reset (POR) and integrated low voltage detect (LVD) with reset (brownout) capability and selectable LVD trip points.
Low-leakage wakeup unit (LLWU)	The LLWU module allows the device to wake from low leakage power modes (LLS and VLLS) through various internal peripheral and external pin sources.
Miscellaneous control module (MCM)	The MCM includes integration logic and embedded trace buffer details.
Crossbar switch (XBS)	The XBS connects bus masters and bus slaves, allowing all bus masters to access different bus slaves simultaneously and providing arbitration among the bus masters when they access the same slave.
Bit manipulation engine (BME)	The BME provides hardware support for atomic read-modify-write memory operations to the peripheral address space in Cortex-M0+ based microcontrollers.
Peripheral bridges	The peripheral bridge converts the crossbar switch interface to an interface to access a majority of peripherals on the device.
DMA multiplexer (DMAMUX)	The DMA multiplexer selects from many DMA requests down to a smaller number for the DMA controller.
Direct memory access (DMA) controller	The DMA controller provides programmable channels with transfer control descriptors for data movement via dual-address transfers for 8-, 16-, 32-, 128-, and 256-bit data values.
External watchdog monitor (EWM)	The EWM is a redundant mechanism to the software watchdog module that monitors both internal and external system operation for fail conditions.
Software watchdog (WDOG)	The WDOG monitors internal system operation and forces a reset in case of failure. It can run from an independent 1 KHz low power oscillator with a programmable refresh window to detect deviations in program flow or system frequency.
Kinetis Flashloader	The Kinetis Flashloader's main task is to load a customer firmware image into the flash memory. This chapter describes Kinetis Flashloader features, functionality, command structure and which peripherals are supported

2.2.3 Memories and Memory Interfaces

The following memories and memory interfaces are available on this device.

Table 2-4. Memories and memory interfaces

Module	Description
Flash memory	Program flash memory — non-volatile flash memory that can execute program code.
Flash memory controller	Manages the interface between the device and the on-chip flash memory.
SRAM	Internal system RAM.

2.2.4 Clocks

The following clock modules are available on this device.

Table 2-5. Clock modules

Module	Description
Multi-clock generator (MCG)	The MCG provides several clock sources for the MCU that include: • Frequency-locked loop (FLL) — Digitally-controlled oscillator (DCO) • Internal reference clocks — Can be used as a clock source for other on-chip peripherals
System oscillator	The system oscillator, in conjunction with an external crystal or resonator, generates a reference clock for the MCU.

2.2.5 Security and Integrity modules

The following security and integrity module is available on this device:

Table 2-6. Security and integrity modules

Module	Description
Cyclic Redundancy Check (CRC)	Hardware CRC generator circuit using 16/32-bit shift register. Error detection for all single, double, odd, and most multi-bit errors, programmable initial seed value, and optional feature to transpose input data and CRC result via transpose register.

2.2.6 Analog modules

The following analog modules are available on this device:

Table 2-7. Analog modules

Module	Description
16-bit analog-to-digital converters (ADC)	16-bit successive-approximation ADC
Analog comparators	Compares two analog input voltages across the full range of the supply voltage.
12-bit digital-to-analog converters (DAC)	Low-power general-purpose DAC, whose output can be placed on an external pin or set as one of the inputs to the analog comparator or ADC.

2.2.7 Timer modules

The following timer modules are available on this device:

Table 2-8. Timer modules

Module	Description
Programmable delay block (PDB)	• 16-bit resolution • 3-bit prescaler • Positive transition of trigger event signal initiates the counter • Supports two triggered delay output signals, each with an independently-controlled delay from the trigger event • Outputs can be OR'd together to schedule two conversions from one input trigger event and can schedule precise edge placement for a pulsed output. This feature is used to generate the control signal for the CMP windowing feature and output to a package pin if needed for applications, such as critical conductive mode power factor correction. • Continuous-pulse output or single-shot mode supported, each output is independently enabled, with possible trigger events • Supports bypass mode • Supports DMA
Flexible timer modules (FTM)	• Selectable FTM source clock, programmable prescaler • 16-bit counter supporting free-running or initial/final value, and counting is up or up-down • Input capture, output compare, and edge-aligned and center-aligned PWM modes • Operation of FTM channels as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs • Deadtime insertion is available for each complementary pair • Generation of hardware triggers • Software control of PWM outputs • Up to 4 fault inputs for global fault control • Configurable channel polarity • Programmable interrupt on input capture, reference compare, overflowed counter, or detected fault condition • Quadrature decoder with input filters, relative position counting, and interrupt on position count or capture of position count on external event • DMA support for FTM events

Table continues on the next page...

Table 2-8. Timer modules (continued)

Module	Description
Low-power timer (LPTimer)	• Selectable clock for prescaler/glitch filter of 1 kHz (internal LPO), 32.768 kHz (external crystal), or internal reference clock • Configurable Glitch Filter or Prescaler with 16-bit counter • 16-bit time or pulse counter with compare • Interrupt generated on Timer Compare • Hardware trigger generated on Timer Compare

2.2.8 Communication interfaces

The following communication interfaces are available on this device:

Table 2-9. Communication modules

Module	Description
Serial peripheral interface (SPI)	Synchronous serial bus for communication to an external device
Inter-integrated circuit (I2C)	Allows communication between a number of devices. Also supports the System Management Bus (SMBus) Specification, version 2.
Universal asynchronous receiver/ transmitters (UART)	Asynchronous serial bus communication interface with programmable 8- or 9-bit data format and support of.
Flex Controller Area Network (FlexCAN)	Supports CAN protocol according to the CAN 2.0 B protocol specification.

2.2.9 Human-machine interfaces

The following human-machine interfaces (HMI) are available on this device:

Table 2-10. HMI modules

Module	Description
General purpose input/output (GPIO)	All general purpose input or output (GPIO) pins are capable of interrupt and DMA request generation. All GPIO pins have 3.3 V tolerance.

2.2.10 Kinetis Motor Suite (KMS)

A selection of KV1x devices are enabled with Kinetis Motor Suite (KMS). The supported devices are listed in Orderable Kinetis Motor Suite part numbers summary. KMS uses the top 8K of flash to store a KMS library file, and is protected using Flash Access Controls. This code space is execute only. For more information refer to Kinetis Motor Suite API Reference Manual (KMSRM)[1] and Kinetis Motor Suite User's Guide (KMSUG)[1].

Table 2-11. KMS module

Module	Description
Kinetis Motor Suite (KMS)	KMS includes firmware preprogrammed on the Kinetis V1x series of microcontrollers and an intuitive PC-based graphical user interface. It supports field oriented control of three phase permanent magnet and brushless DC motors for sensorless velocity control and sensored position control.

KMS part numbers for different motor types:

- P suffix for sensorless or sensored FOC for velocity control of PMSM and BLDC motors

2.3 Orderable part numbers

The following table summarizes the part numbers of the devices covered by this document.

Table 2-12. Orderable part numbers summary

NXP part number	CPU frequency	Pin count	Package	Total flash memory	FlexCAN	SRAM	GPIO
MKV11Z128VLH7	75 MHz	64	LQFP	128 KB	Yes	16 KB	46
MKV11Z128VLF7	75 MHz	48	LQFP	128 KB	Yes	16 KB	35
MKV11Z128VLC7[1]	75 MHz	32	LQFP	128 KB	Yes	16 KB	26
MKV11Z128VFM7	75 MHz	32	QFN	128 KB	Yes	16 KB	26
MKV11Z64VLH7	75 MHz	64	LQFP	64 KB	Yes	16 KB	46
MKV11Z64VLF7	75 MHz	48	LQFP	64 KB	Yes	16 KB	35
MKV11Z64VLC7[1]	75 MHz	32	LQFP	64 KB	Yes	16 KB	26
MKV11Z64VFM7	75 MHz	32	QFN	64 KB	Yes	16 KB	26
MKV10Z64VLH7	75 MHz	64	QFP	64 KB	No	16 KB	26
MKV10Z64VLF7	75 MHz	48	QFP	64 KB	No	16 KB	26
MKV10Z64VLC7 [1]	75 MHz	32	LQFP	64 KB	No	16 KB	26
MKV10Z64VFM7	75 MHz	32	QFN	64 KB	No	16 KB	26
MKV10Z128VLH7	75 MHz	64	QFP	128 KB	No	16 KB	46
MKV10Z128VLF7	75 MHz	48	QFP	128 KB	No	16 KB	35
MKV10Z128VLC7[1]	75 MHz	32	LQFP	128 KB	No	16 KB	26
MKV10Z128VFM7	75 MHz	32	QFN	128 KB	No	16 KB	26

1. The 32-pin LQFP package supporting this part number is not yet available, however it is included in a Package Your Way program for Kinetis MCUs. Please visit http://www.nxp.com/KPYW for more details

1. To find the associated resource, go to http://www.nxp.com and perform a search using Document ID

Table 2-13. KMS orderable part numbers summary

NXP part number	Motor type	Pin count	Package	Program flash[1]	SRAM	GPIO
MKV11Z128VLH7P	PMSM / BLDC	64	LQFP	120 KB	16 KB	46
MKV11Z128VLF7P	PMSM / BLDC	48	LQFP	120 KB	16 KB	35
MKV11Z128VLC7P[2]	PMSM / BLDC	32	LQFP	120 KB	16 KB	26
MKV10Z64VLH7P	PMSM / BLDC	64	LQFP	56 KB	16 KB	46
MKV10Z64VLF7P	PMSM / BLDC	48	LQFP	56 KB	16 KB	35
MKV10Z64VLC7P [2]	PMSM/ BLDC	32	LQFP	56 KB	16 KB	26
MKV11Z128VFM7P	PMSM/BLDC	32	QFN	120 KB	16 KB	26
MKV10Z64VFM7P	PMSM/BLDC	32	QFN	56 KB	16 KB	26

1. Kinetis Motor Suite enabled devices have 8K bytes reserved flash space as execute only. See Kinetis Motor Suite Configuration
2. The 32-pin LQFP package supporting this part number is not yet available, however it is included in a Package Your Way program for Kinetis MCUs. Please visit http://www.nxp.com/KPYW for more details

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
- Links for more information

3.2 Module to Module Interconnects

3.2.1 Module to Module interconnections

The table below shows the various signal interconnections for MKV1x devices.

Table 3-1. MKV1x peripheral interconnections

Peripheral	Signal	Signal name	In or Out	connected to
LPTMR0	TPS [5:4]	—	—	—
—	0;0	Pulse counter input 0	input	CMP0_OUT
—	0;1	Pulse counter input 1	input	LPTMR0_ALT1 pin
—	1;0	Pulse counter input 2	input	LPTMR0_ALT2 pin
—	1;1	Pulse counter input 3	input	LPTMR0_ALT3 pin
—	PCS[1:0]	—	—	—
—	0;0	Prescaler/Glitchfilter input clock 0	input	MCGIRCLK
—	0;1	Prescaler/Glitchfilter input clock 1	input	1khz LPO

Table continues on the next page...

Table 3-1. MKV1x peripheral interconnections (continued)

Peripheral	Signal	Signal name	In or Out	connected to
—	1;0	Prescaler/Glitchfilter input clock 2	input	OSC32KCLK
—	1;1	Prescaler/Glitchfilter input clock 2	input	OSCERCLK
FTM0	TRIG0	Triggers refresh of FTM regs/counter	input	FTM0SYNCBIT of SIM_SOPT8 reg/ CMP0_OUT or FTM1_match
—	TRIG1	Triggers refresh of FTM regs/counter	input	PDB0 ch1 trigger or FTM2_match
—	TRIG2	Triggers refresh of FTM regs/counter	input	CMP0_out or CMP1_out
—	FAULT0	places PWMs in safe state	input	FTM0_FLT0 or CMP0_OUT
—	FAULT1	places PWMs in safe state	input	FTM0_FLT1 or CMP1_OUT
—	FAULT2	places PWMs in safe state	input	FTM0_FLT2
—	FAULT3	places PWMs in safe state	input	FTM0_FLT3
—	EXTRIG	OR of all channels/ counter init	output	PDB0/PDB1 ch 1000
—	Ch0 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 24
—	Ch1 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 25
—	Ch2 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 26
—	Ch3 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 27
—	Ch4 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 28
—	Ch5 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 29
	FTM0_init trigger	counter overflow	output	PDB0 /PDB1 ch 1000
FTM1	TRIG0	Triggers refresh of FTM regs/counter	input	FTM1SYNCBIT of SIM_SOPT8 reg/ CMP0_OUT or FTM0_match
—	TRIG1	Triggers refresh of FTM regs/counter	input	PDB0 ch1 trigger or FTM2_match
—	TRIG2	Triggers refresh of FTM regs/counter	input	CMP0_out or CMP1_out
—	FAULT0	places PWMs in safe state	input	FTM1_FLT0 or CMP0_OUT
—	FAULT1	places PWMs in safe state	input	CMP1_OUT
—	FTM1CH0SRC	input option for channel 0 capture	input	FTM1_CH0 pin or CMP0_OUT or CMP1_OUT

Table continues on the next page...

Table 3-1. MKV1x peripheral interconnections (continued)

Peripheral	Signal	Signal name	In or Out	connected to
—	EXTRIG	OR of all channels/ counter init	output	PDB0/PDB1 ch 1001
—	Ch0 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 32
—	Ch0 output	Channel event (CMP/ CAP)	output	UART0TXSRC/UART1TXSRC
—	Ch1 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 33
—	Ch1 output	Channel event (CMP/ CAP)	output	UART0TXSRC/UART1TXSRC
—	FTM1_init trigger	counter overflow	output	PDB0/PDB1 ch 1001
—	PHA	input A for quad decode	input	FTM1_QD_PHA
—	PHB	input B for quad decode	input	FTM1_QD_PHB
FTM2	TRIG0	Triggers refresh of FTM regs/counter	input	FTM2SYNCBIT of SIM_SOPT8 reg/ CMP0_OUT or FTM0_match
—	TRIG1	Triggers refresh of FTM regs/counter	input	PDB0 ch1 trigger or FTM1_match
—	TRIG2	Triggers refresh of FTM regs/counter	input	CMP0_out or CMP1_out
—	FAULT0	places PWMs in safe state	input	FTM2_FLT0 or CMP0_OUT
—	FAULT1	places PWMs in safe state	input	CMP1_OUT
—	FTM2CH0SRC	input option for channel 0 capture	input	FTM2_CH0 pin or CMP0_OUT or CMP1_OUT
—	EXTRIG	OR of all channels/ counter init	output	PDB0/PDB1 ch 1010
—	Ch0 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 34
—	Ch1 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 35
—	FTM2_init trigger	counter overflow	input	PDB0/PDB1 ch 1010
—	PHA	input A for quad decode	input	FTM2_QD_PHA
—	PHB	input B for quad decode	input	FTM2_QD_PHB
FTM3	TRIG0	Triggers refresh of FTM regs/counter	input	FTM3SYNCBIT of SIM_SOPT8 reg/ CMP0_OUT or FTM5_match
—	TRIG1	Triggers refresh of FTM regs/counter	input	PDB1 ch1 trigger or FTM4_match
—	TRIG2	Triggers refresh of FTM regs/counter	input	CMP0_out or CMP1_out
—	FAULT0	places PWMs in safe state	input	FTM3_FLT0 or CMP0_OUT
—	FAULT1	places PWMs in safe state	input	CMP1_OUT

Table continues on the next page...

Table 3-1. MKV1x peripheral interconnections (continued)

Peripheral	Signal	Signal name	In or Out	connected to
—	FAULT2	places PWMs in safe state	input	tied to 0
—	FAULT3	places PWMs in safe state	input	tied to 0
—	EXTRG	OR of all channels/ counter init	input	PDB0/PDB1 ch 1011
—	Ch0 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 36
—	Ch1 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 37
—	Ch2 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 38
—	Ch3 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 39
—	Ch4 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 54
—	Ch5 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 55
—	FTM3_init trigger	counter overflow	input	PDB0/PDB1 ch 1011
FTM4	TRIG0	Triggers refresh of FTM regs/counter	input	FTM4SYNCBIT of SIM_SOPT8 reg/ CMP0_OUT or FTM3_match
—	TRIG1	Triggers refresh of FTM regs/counter	input	PDB1 ch1 trigger or FTM5_match
—	TRIG2	Triggers refresh of FTM regs/counter	input	CMP0_out or CMP1_out
—	FAULT0	refresh PWMs or output CMPs	input	FTM4_FLT0 or CMP0_OUT
—	FAULT1	refresh PWMs or output CMPs	input	CMP1_OUT
—	FTM4CH0SRC	Triggers refresh of FTM regs/counter	input	FTM4_CH0 pin or CMP0_OUT or CMP1_OUT
—	EXTRIG	Triggers refresh of FTM regs/counter	output	PDB0/PDB1 ch 1100
—	Ch0 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 30
—	Ch1 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 31
—	FTM4_init trigger	counter overflow	input	PDB0/PDB1 ch 1100
—	PHA	input A for quad decode	input	FTM4_CH0
—	PHB	input B for quad decode	input	FTM4_CH1
FTM5	TRIG0	Triggers refresh of FTM regs/counter	input	FTM5SYNCBIT of SIM_SOPT8 reg/ CMP0_OUT or FTM3_match
—	TRIG1	Triggers refresh of FTM regs/counter	input	PDB1 ch1 trigger or FTM4_match

Table continues on the next page...

Table 3-1. MKV1x peripheral interconnections (continued)

Peripheral	Signal	Signal name	In or Out	connected to
—	TRIG2	Triggers refresh of FTM regs/counter	input	CMP0_out or CMP1_out
—	FAULT0	refresh PWMs or output CMPs	input	FTM5_FLT0 or CMP0_OUT
—	FAULT1	refresh PWMs or output CMPs	input	CMP1_OUT
—	FTM5CH0SRC	input option for channel 0 capture	input	FTM5_CH0 pin or CMP0_OUT or CMP1_OUT
—	EXTRIG	OR of all channels/ counter init	output	PDB0/PDB1 ch 1101
—	Ch0 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 56
—	Ch1 event ipd_req	Channel event (CMP/ CAP)	output	DMA_MUX source 57
—	FTM5_init trigger	counter overflow	input	PDB0/PDB1 ch 1101
—	PHA	input A for quad decode	input	FTM5_QD_PHA
—	PHB	input B for quad decode	input	FTM5_QD_PHB
PDB0	0	input 0	input	External Trigger (PDB_EXTRG0) (PTC0 pin)
—	1	input 1	input	CMP0_OUT
—	10	input 2	input	CMP1_OUT
—	11	input	input	PDB_EXTRG1 (PTC6 pin)
—	100	—	input	DMAch0_done
—	101	—	input	DMAch1_done
—	110	—	input	DMAch2_done
—	111	—	input	DMAch3_done
—	1000	—	input	FTM0 Init and Ext Trigger Outputs
—	1001	—	input	FTM1 Init and Ext Trigger Outputs
—	1010	—	input	FTM2 init and Ext Trigger Outputs
—	1011	—	input	FTM3 init and Ext Trigger Outputs
—	1100	—	input	FTM4 init and Ext Trigger Outputs
—	1101	—	input	FTM5 init and Ext Trigger Outputs
—	1110	input 15	input	LPTMR Output
—	1111	software triggered input	—	—
—	ch0 trigger	trigger 0 out	output	ADC0 (via SIM_SOPT7)
	ch0_pre_trigger	pre-trigger_out0	output	ADC0 pre-trigger
	ipd_req_trigger DMA		output	DMA_MUX source48
—	ch1 trigger	trigger 1 out	output	ADC1 (via SIM_SOPT7) and TRIG1 of FTM0,1,2
—	ch1_pre_trigger	pre-trigger_out1	output	ADC1 pre-trigger
—	ch0 ACK	channel 0 acknowledge	input	ADC0COCO[0] and ADC1COCO[1]

Table continues on the next page...

Table 3-1. MKV1x peripheral interconnections (continued)

Peripheral	Signal	Signal name	In or Out	connected to
—	ch1 ACK	channel1 acknowledge	input	ADC0COCO[1] and ADC1COCO[0]
—	pulse out	—	output	each CMP sample/window input
—	DAC triggers	—	output	DAC0 trigger
PDB1	0	input 0	input	External Trigger (PDB_EXTRG0) (PTC0pin)
—	1	input 1	input	CMP0_OUT
—	10	input 2	input	CMP1_OUT
—	11	input	input	External Trigger (PDB_EXTRG1) (PTC6 pin)
—	100	—	input	DMAch4_done
—	101	—	input	DMAch5_done
—	110	—	input	DMAch6_done
—	111	—	input	DMAch7_done
—	1000	—	input	FTM0 Init and Ext Trigger Outputs
—	1001	—	input	FTM1 Init and Ext Trigger Outputs
—	1010	—	input	FTM2Init and Ext Trigger Outputs
—	1011	—	input	FTM3 Init and Ext Trigger Outputs
—	1100	—	input	FTM4 Init and Ext Trigger Outputs
—	1101	—	input	FTM5 Init and Ext Trigger Outputs
—	1110	input 15	input	LPTMR Output
—	1111	software triggered input	—	—
—	ch0 trigger	trigger 0 out	output	ADC0 (via SIM_SOPT7)
—	ch0_pre_trigger	pre-trigger_out0	output	ADC0 pre-trigger
—	ipd_req_trigger DMA		output	DMA_MUX source47
—	ch1 trigger	trigger 1 out	output	ADC1 (via SIM_SOPT7) and TRIG1 of FTM3, FTM4, FTM5
	ch1_pre_trigger	pre-trigger_out1	output	ADC1 pre-trigger
—	ch0 ACK	channel 0 acknowledge	input	ADC0COCO[0] and ADC1COCO[1]
—	ch1 ACK	channel1 acknowledge	input	ADC0COCO[1] and ADC1COCO[0]
—	pulse out	—	output	each CMP sample/window input
—	DAC triggers	—	output	DAC0 trigger
Watchdog	CLK_IN0	input clock source default	input	LPO or MCGIRCLK
—	CLK_IN1	alternative clock source	input	Bus clock
—	CLK_IN3	fast test clock source	input	Bus clock
ADC0	ch0	analog input channel	input	ADC0_SE0/ADC0_DP0 and ADC0_DM0
—	ch1	analog input channel	input	ADC0_SE1/ADC0_DP1 and ADC0_DM1
—	ch2	analog input channel	input	ADC0_SE2

Table continues on the next page...

Table 3-1. MKV1x peripheral interconnections (continued)

Peripheral	Signal	Signal name	In or Out	connected to
—	ch3	analog input channel	input	ADC0_SE3
—	ch4	analog input channel	input	ADC0_SE4
—	ch5	analog input channel	input	ADC0_SE5
—	ch6	analog input channel	input	ADC0_SE6
—	ch7	analog input channel	input	ADC0_SE7
—	ch8	analog input channel	input	ADC0_SE8
—	ch9	analog input channel	input	ADC0_SE9
—	ch10	analog input channel	input	ADC0_SE10
—	ch11	analog input channel	input	ADC0_SE11
—	ch12	analog input channel	input	ADC0_SE12
—	ch13	analog input channel	input	ADC0_SE13
—	ch14	analog input channel	input	ADC0_SE14
—	ch25	analog input channe	input	VREFH
—	ch26	analog input channe	input	Temperature sensor
—	ch27	analog input channe	input	Bandgap
—	hardware trigger	trigger input, start acquisition	input	PDB0_out 0 or PDB1_out 0 via SIM_SOPT7 bit
—	COCO	conversion complete	output	DMA _MUX source 40, PDB0 channel 0 ACK, PDB0 channel 1 ACK, PDB1 channel 0 ACK, PDB1 channel 1 ACK
ADC1	ch0	analog input channel	input	ADC1_SE0
—	ch1	analog input channel	input	ADC1_SE1/ADC1_DP1 and ADC1_DM1
—	ch2	analog input channel	input	ADC1_SE2/ADC1_DP2 and ADC1_DM2
—	ch3	analog input channel	input	ADC1_SE3
—	ch4	analog input channel	input	ADC1_SE4
—	ch5	analog input channel	input	ADC1_SE5
—	ch6	analog input channel	input	ADC1_SE6
—	ch7	analog input channel	input	ADC1_SE7
—	ch8	analog input channel	input	ADC1_SE8
—	ch9	analog input channel	input	ADC1_SE9
—	ch10	analog input channel	input	ADC1_SE10
—	ch11	analog input channel	input	ADC1_SE11
—	ch12	analog input channel	input	ADC1_SE12
—	ch13	analog input channel	input	ADC1_SE13
—	ch14	analog input channel	input	ADC1_SE14
—	ch15	analog input channel	input	ADC1_SE15
—	ch16	analog input channel	input	ADC1_SE16
—	ch17	analog input channel	input	ADC1_SE17

Table continues on the next page...

Table 3-1. MKV1x peripheral interconnections (continued)

Peripheral	Signal	Signal name	In or Out	connected to
—	ch25	analog input channel	input	VREFH
—	ch26	analog input channel	input	Temperature sensor
—	ch27	analog input channel	input	Bandgap
—	hardware trigger	trigger input, start acquisition	input	PDB0_out 1 or PDB1_out 1 via SIM_SOPT7 bit
—	COCO	conversion complete	output	DMA_MUX source 41, PDB0 channel 0 ACK, PDB0 channel 1 ACK, PDB1 channel 0 ACK, PDB1 channel 1 ACK
CMP0	IN0	input channel	input	CMP0_IN0 pin
—	IN1	input channel	input	CMP0_IN1 pin
—	IN2	input channel	input	CMP0_IN2 pin
—	IN3	input channel	input	CMP0_IN3 pin
—	IN4	input channel	input	CMP0_IN4 pin
—	IN5	input channel	input	CMP0_IN5 pin
—	IN6	input channel	input	bandgap
—	IN7	input channel	input	6bDAC
—	WindowIN	—	input	PDB0 or PDB1 pulse output
—	CMP0_OUT	—	output	FTM0TRG0SRC, FTM0TRG2SRC, FTM0FLT0, FTM1TRG0SRC, FTM1TRG2SRC, FTM1FLT0, FTM1CH0SRC, FTM2TRG0SRC, FTM2TRG2SRC, FTM2FLT0, FTM2CH0SRC, UART0RXSRC, FTM3TRG0SRC, FTM3TRG2SRC, FTM3FLT0, FTM4TRG0SRC, FTM4TRG2SRC, FTM4FLT0, FTM4CH0SRC, FTM5TRG0SRC, FTM5TRG2SRC, FTM5FLT0, FTM5CH0SRC, UART1RXSRC, ADC0TRGSEL, and ADC1TRGSEL, (all via SIM_SOPTx registers) LLWU_M1IF, DMAMUX source 42, PDB0 input 0001, PDB1 input 0001, LPTMR0
CMP1	IN0	input channel	input	CMP1_IN0 pin
—	IN1	input channel	input	CMP1_IN1 pin
—	IN2	input channel	input	—
—	IN3	input channel	input	—
—	IN4	input channel	input	CMP1_IN4
—	IN5	input channel	input	CMP1_IN5
—	IN6	input channel	input	bandgap
—	IN7	input channel	input	6bDAC
—	WindowIN	—	input	PDB0 or PDB1 pulse output

Table continues on the next page...

Table 3-1. MKV1x peripheral interconnections (continued)

Peripheral	Signal	Signal name	In or Out	connected to
—	CMP1_OUT	—	output	FTM0TRG2SRC, FTM0FLT1, FTM1TRG2SRC, FTM1_FAULT1, FTM2TRG2SRC, FTM2_FAULT1, FTM3TRG2SRC, FTM3FLT1, FTM4TRG2SRC, FTM4_FAULT1, FTM5TRG2SRC, FTM5_FAULT1, ADC0TRGSEL, and ADC1TRGSEL (all via SIM_SOPTx registers), LLWU_M2IF, PDB0 input 0010, PDB1 input 0010
UART0	Receive complete	—	output	DMA _MUXsource 2
—	Transmit complete	—	output	DMA _MUXsource 3
	UART0RXSRC	—	input	UART0_RX pin or CMP0_OUT or CMP1_OUT
	UART0TXSRC	—	output	UART0_TX pin or UART0_TX pin modulated with FTM1 channel 0 output or FTM2 channel 0 output
UART1	Receive complete	—	output	DMA _MUXsource 4
—	Transmit complete	—	output	DMA_MUX source 5
	UART1RXSRC	—	input	UART1_RX pin or CMP0_OUT or CMP1_OUT
	UART1TXSRC	—	output	UART1_TX pin or UART1_TX pin modulated with FTM1 channel 0 output or FTM2 channel 0 output
SPI	Receive complete	—	output	DMA_MUX source 16
—	Transmit complete	—	output	DMA_MUX source 17
I2C	transmission complete	—	output	DMA_MUX source 22
FlexCAN0	Receive full	—	output	DMA_MUX source 14

3.3 Core Modules

3.3.1 ARM Cortex-M0+ Core Configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at **arm.com**.

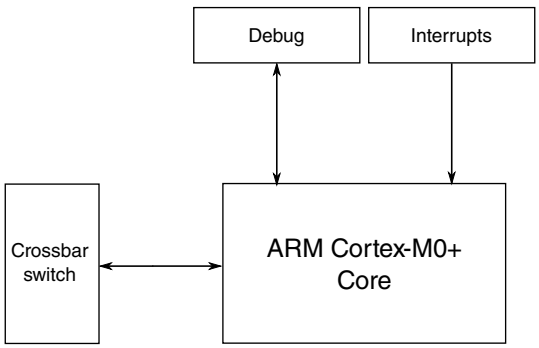


Figure 3-1. Core configuration

Table 3-2. Reference links to related information

Topic	Related module	Reference
Full description	ARM Cortex-M0+ core, r0p1	ARM Cortex-M0+ Technical Reference Manual, r0p1
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
System/instruction/data bus module	Crossbar switch	Crossbar switch
Debug	Serial Wire Debug (SWD)	Debug
	Micro Trace Buffer (MTB)	MTB
Interrupts	Nested Vectored Interrupt Controller (NVIC)	NVIC
	Miscellaneous Control Module (MCM)	MCM
	Memory-Mapped Divide and Square Root (MMDVSQ)	MMDVSQ

3.3.1.1 ARM Cortex M0+ Core

The ARM Cortex M0+ parameter settings are as follows:

Table 3-3. ARM Cortex-M0+ parameter settings

Parameter	Verilog Name	Value	Description
Arch Clock Gating	ACG	1 = Present	Implements architectural clock gating
DAP Slave Port Support	AHBSLV	1	Support any AHB debug access port (like the CM0+ DAP)

Table continues on the next page...

Table 3-3. ARM Cortex-M0+ parameter settings (continued)

Parameter	Verilog Name	Value	Description
DAP ROM Table Base	BASEADDR	0xF000_2003	Base address for DAP ROM table
Endianess	BE	0	Little endian control for data transfers
Breakpoints	BKPT	2	Implements 2 breakpoints
Debug Support	DBG	1 = Present	
Halt Event Support	HALTEV	1 = Present	
I/O Port	IOP	1 = Present	Implements single-cycle ld/st accesses to special address space
IRQ Mask Enable	IRQDIS	0x00000000	All 32 IRQs are used (set if IRQ is disabled)
Debug Port Protocol	JTAGnSW	0 = SWD	SWD protocol, not JTAG
Core Memory Protection	MPU	0 = Absent	No MPU
Number of IRQs	NUMIRQ	32	Full NVIC request vector
Reset all regs	RAR	0 = Standard	Do not force all registers to be async reset
Multiplier	SMUL	0 = Fast Mul	Implements single-cycle multiplier
Multi-drop Support	SWMD	0 = Absent	Do not include serial wire support for multi-drop
System Tick Timer	SYST	1 = Present	Implements system tick timer
DAP Target ID	TARGETID	0	
User/Privileged	USER	1 = Present	Implements processor operating modes
Vector Table Offset Register	VTOR	1 = Present	Implements relocation of exception vector table
WIC Support	WIC	1 = Present	Implements WIC interface
WIC Requests	WICLINES	34	Exact number of wakeup IRQs is 34
Watchpoints	WPT	2	Implements 2 watchpoints

For details on the ARM Cortex-M0+ processor core, see the ARM website: www.arm.com.

3.3.1.2 Buses, Interconnects, and Interfaces

The ARM Cortex-M0+ core has two bus interfaces:

- single 32-bit AMBA-3 AHB-Lite system interface that provides connections to peripherals and all system memory, which includes flash and RAM.
- single 32-bit I/O port bus interfacing to the GPIO with 1-cycle loads and stores.

3.3.1.3 System Tick Timer

The CLKSOURCE bit in SysTick Control and Status register selects either the core clock (when CLKSOURCE = 1) or a divide-by-16 of the core clock (when CLKSOURCE = 0). Because the timing reference is a variable frequency, the TENMS bit in the SysTick Calibration Value Register is always zero.

3.3.1.4 Debug Facilities

This device supports standard ARM 2-pin SWD debug port.

3.3.1.5 Core Privilege Levels

The Core on this device is implemented with both Privileged and Unprivileged levels. The ARM documentation uses different terms than this document to distinguish between privilege levels.

If you see this term...	it also means this term...
Privileged	Supervisor
Unprivileged or user	User

3.3.2 Nested Vectored Interrupt Controller (NVIC) Configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at **arm.com**.

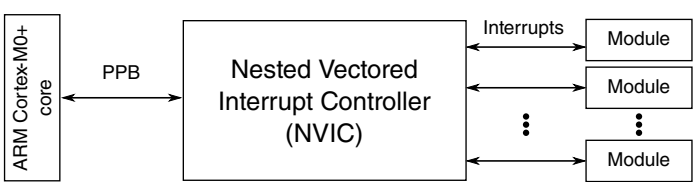


Figure 3-2. NVIC configuration

Table 3-4. Reference links to related information

Topic	Related module	Reference
Full description	Nested Vectored Interrupt Controller (NVIC)	ARM Cortex-M0+ Technical Reference Manual
System memory map		System memory map
Clocking		Clock distribution

Table continues on the next page...

Table 3-4. Reference links to related information (continued)

Topic	Related module	Reference
Power management		Power management
Private Peripheral Bus (PPB)	ARM Cortex-M0+ core	ARM Cortex-M0+ core

3.3.2.1 Interrupt priority levels

This device supports 4 priority levels for interrupts. Therefore, in the NVIC each source in the IPR registers contains 2 bits. For example, IPR0 is shown below:

	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	IRQ3		0	0	0	0	0	0	IRQ2		0	0	0	0	0	0	IRQ1		0	0	0	0	0	0	IRQ0		0	0	0	0	0	0
W																																

3.3.2.2 Non-maskable interrupt

The non-maskable interrupt request to the NVIC is controlled by the external $\overline{\text{NMI}}$ signal. The pin the $\overline{\text{NMI}}$ signal is multiplexed on, must be configured for the $\overline{\text{NMI}}$ function to generate the non-maskable interrupt request.

3.3.2.3 Interrupt channel assignments

The interrupt vector assignments are defined in the following table.

- Vector number — the value stored on the stack when an interrupt is serviced.
- IRQ number — non-core interrupt source count, which is the vector number minus 16.

The IRQ number is used within ARM's NVIC documentation.

Table 3-6. Interrupt vector assignments

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
ARM Core System Handler Vectors					
0x0000_0000	0	—	—	ARM core	Initial Stack Pointer
0x0000_0004	1	—	—	ARM core	Initial Program Counter

Table continues on the next page...

Table 3-6. Interrupt vector assignments (continued)

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
0x0000_0008	2	—	—	ARM core	Non-maskable Interrupt (NMI)
0x0000_000C	3	—	—	ARM core	Hard Fault
0x0000_0010	4	—	—	—	—
0x0000_0014	5	—	—	—	—
0x0000_0018	6	—	—	—	—
0x0000_001C	7	—	—	—	—
0x0000_0020	8	—	—	—	—
0x0000_0024	9	—	—	—	—
0x0000_0028	10	—	—	—	—
0x0000_002C	11	—	—	ARM core	Supervisor call (SVCall)
0x0000_0030	12	—	—	—	—
0x0000_0034	13	—	—	—	—
0x0000_0038	14	—	—	ARM core	Pendable request for system service (PendableSrvReq)
0x0000_003C	15	—	—	ARM core	System tick timer (SysTick)
Non-Core Vectors					
0x0000_0040	16	0	0	DMA	DMA channel 0, 4 transfer complete
0x0000_0044	17	1	0	DMA	DMA channel 1, 5 transfer complete
0x0000_0048	18	2	0	DMA	DMA channel 2, 6 transfer complete
0x0000_004C	19	3	0	DMA	DMA channel 3, 7 transfer complete
0x0000_0050	20	4	1	DMA	DMA errors
0x0000_0054	21	5	1	FTFA	Command complete and read collision
0x0000_0058	22	6	1	PMC	Low-voltage detect, low-voltage warning
0x0000_005C	23	7	1	LLWU	Low Leakage Wakeup
0x0000_0060	24	8	2	I^2C0	—
0x0000_0064	25	9	2	—	—
0x0000_0068	26	10	2	SPI0	Single interrupt vector for all sources
0x0000_006C	27	11	2	—	
0x0000_0070	28	12	3	UART0	Status and error
0x0000_0074	29	13	3	UART1	Status and error
0x0000_0078	30	14	3	FlexCAN0	
0x0000_007C	31	15	3	ADC0	—
0x0000_0080	32	16	4	ADC1	—
0x0000_0084	33	17	4	FTM0	Single interrupt vector for all sources
0x0000_0088	34	18	4	FTM1	Single interrupt vector for all sources
0x0000_008C	35	19	4	FTM2	Single interrupt vector for all sources
0x0000_0090	36	20	5	CMP0	—
0x0000_0094	37	21	5	CMP1	—

Table continues on the next page...

Table 3-6. Interrupt vector assignments (continued)

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
0x0000_0098	38	22	5	FTM3	—
0x0000_009C	39	23	5	WDOG/EWM	—
0x0000_00A0	40	24	6	FTM4	—
0x0000_00A4	41	25	6	DAC0	—
0x0000_00A8	42	26	6	FTM5	—
0x0000_00AC	43	27	6	MCG	—
0x0000_00B0	44	28	7	LPTMR0	—
0x0000_00B4	45	29	7	PDB0 and PDB1	—
0x0000_00B8	46	30	7	Port control module	Pin detect (Port A)
0x0000_00BC	47	31	7	Port control module	Pin detect (Port B, C, D, and E)

1. Indicates the NVIC's interrupt source number.
2. Indicates the NVIC's IPR register number used for this IRQ. The equation to calculate this value is: IRQ div 4

3.3.2.3.1 Determining the bitfield and register location for configuring a particular interrupt

Suppose you need to configure the SPI0 interrupt. The following table is an excerpt of the SPI0 row from Interrupt channel assignments.

Table 3-7. Interrupt vector assignments

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
0x0000_0068	26	10	2	SPI0	Single interrupt vector for all sources

1. Indicates the NVIC's interrupt source number.
2. Indicates the NVIC's IPR register number used for this IRQ. The equation to calculate this value is: IRQ div 4.

- The NVIC registers you would use to configure the interrupt are:
  - NVICIPR2
- To determine the particular IRQ's bitfield location within these particular registers:
  - NVICIPR2 bitfield starting location = 8 * (IRQ mod 4) + 6 = 22

  Since the NVICIPR bitfields are 2-bit wide (4 priority levels), the NVICIPR2 bitfield range is 22-23

Therefore, the following bitfield locations are used to configure the SPI0 interrupts:

- NVICIPR2[23:22]

3.3.3 Asynchronous wake-up interrupt controller (AWIC) configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at **arm.com**.

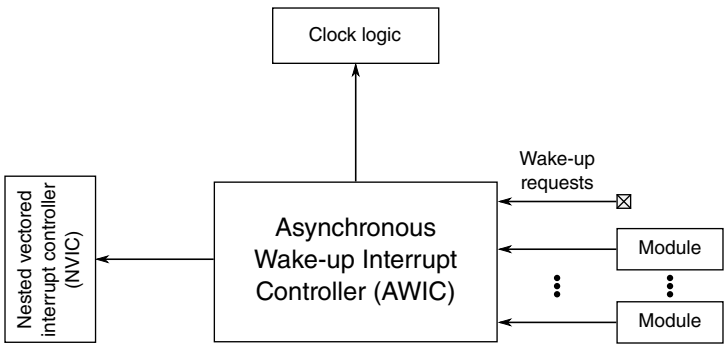


Figure 3-3. Asynchronous wake-up interrupt controller configuration

Table 3-8. Reference links to related information

Topic	Related module	Reference
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
	Nested vectored interrupt controller (NVIC)	NVIC
Wake-up requests		AWIC wake-up sources

3.3.3.1 Wake-up sources

The device uses the following internal and external inputs to the AWIC module.

Table 3-9. AWIC stop wake-up sources

Wake-up source	Description
Available system resets	$\overline{\text{RESET}}$ pin and WDOG when LPO is its clock source
Low-voltage detect	Power management controller - functional in Stop mode
Low-voltage warning	Power management controller - functional in Stop mode
Pin interrupts	Port control module - any enabled pin interrupt is capable of waking the system
ADCx	The ADC is functional when using internal clock source
CMP	Interrupt in normal or trigger mode

Table continues on the next page...

Table 3-9. AWIC stop wake-up sources (continued)

Wake-up source	Description
I^2Cx	Address match wakeup
UART	Active edge on RxD
NMI	NMI pin
LPTMR	Any interrupt provided clock remains enabled
SPI	Slave mode interrupt
MCG	Multi-clock source interrupt
FlexCAN	CAN bus activates wake-up in Stop mode

3.4 System Modules

3.4.1 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.

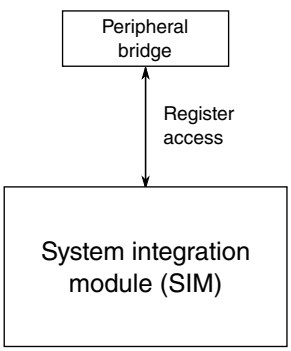


Figure 3-4. SIM configuration

Table 3-10. Reference links to related information

Topic	Related module	Reference
Full description	SIM	SIM
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management

3.4.2 Flashloader 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.

Table 3-11. Reference links to related information

Topic	Related module	Reference
Full description	Kinetis Flashloader	Flashloader
System memory map		Memory map
Clocking		Clock configuration
Peripherals		Supported Peripherals

This device has various peripherals (UART, I2C, SPI,CAN) supported by the Kinetis Flashloader. The following table shows the pads used by the Kinetis Flashloader.

Table 3-12. Kinetis Flashloader Peripheral Pinmux

Port	Signal
PTD6	UART0_RX
PTD7	UART0_TX
PTB0	I2C0_SCL
PTB1	I2C0_SDA
PTE16	SPI0_PCS0
PTE17	SPI0_SCK
PTE18	SPI0_SOUT
PTE19	SPI0_SIN
PTE24	CAN0_Tx
PTE25	CAN0_Rx

3.4.3 System Mode Controller (SMC) 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.

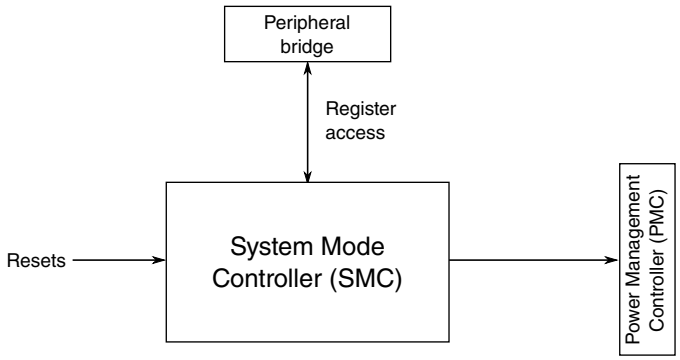


Figure 3-5. System Mode Controller configuration

Table 3-13. Reference links to related information

Topic	Related module	Reference
Full description	System Mode Controller (SMC)	SMC
System memory map		System memory map
Power management		Power management
	Power management controller (PMC)	PMC
	Low-Leakage Wakeup Unit (LLWU)	LLWU
	Reset Control Module (RCM)	Reset

3.4.3.1 VLLS2 not supported

VLLS2 power mode is not supported on this device.

3.4.3.2 LLS not supported

LLS power mode is not supported on this device.

3.4.4 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.

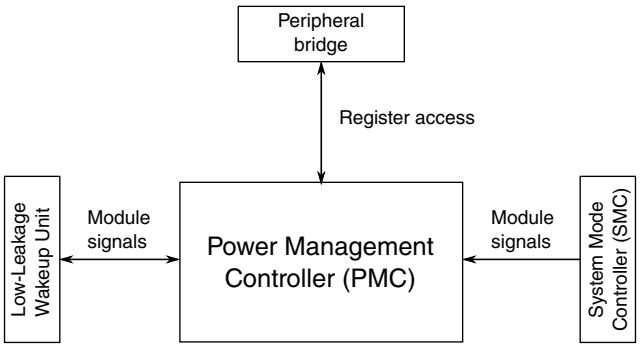


Figure 3-6. PMC configuration

Table 3-14. Reference links to related information

Topic	Related module	Reference
Full description	PMC	PMC
System memory map		System memory map
Power management		Power management
Full description	System Mode Controller (SMC)	System Mode Controller
	Low-Leakage Wakeup Unit (LLWU)	LLWU
	Reset Control Module (RCM)	Reset

3.4.5 Low-Leakage Wake-up Unit (LLWU) 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.

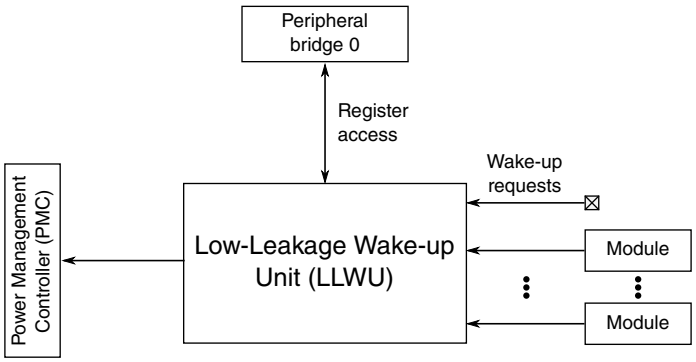


Figure 3-7. Low-Leakage Wake-up Unit configuration

Table 3-15. Reference links to related information

Topic	Related module	Reference
Full description	LLWU	LLWU
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management chapter
	Power Management Controller (PMC)	Power Management Controller (PMC)
	System Mode Controller (SMC)	System Mode Controller
Wake-up requests		LLWU wake-up sources

3.4.5.1 Wake-up Sources

The device uses the following internal peripheral and external pin inputs as wakeup sources to the LLWU module. LLWU_Px are external pin inputs, and LLWU_M0IF-M7IF are connections to the internal peripheral interrupt flags.

NOTE

In addition to the LLWU wakeup sources, the device also wakes from low power modes when NMI or RESET pins are enabled and the respective pin is asserted.

Table 3-16. Wakeup Sources

LLWU pin	Module source or pin name
LLWU_P0	PTE1
LLWU_P1	Reserved
LLWU_P2	Reserved
LLWU_P3	PTA4

Table continues on the next page...

Table 3-16. Wakeup Sources (continued)

LLWU pin	Module source or pin name
LLWU_P4	PTA13
LLWU_P5	PTB0
LLWU_P6	PTC1
LLWU_P7	PTC3
LLWU_P8	PTC4
LLWU_P9	PTC5
LLWU_P10	PTC6
LLWU_P11	PTC11
LLWU_P12	PTD0
LLWU_P13	PTD2
LLWU_P14	PTD4
LLWU_P15	PTD6
LLWU_P16	Reserved
LLWU_P17	Reserved
LLWU_P18	Reserved
LLWU_P19	PTE17
LLWU_P20	PTE18
LLWU_P21	PTE25
LLWU_P22	Reserved
LLWU_P23	Reserved
LLWU_M0IF	LPTMR0
LLWU_M1IF	CMP0
LLWU_M2IF	CMP1
LLWU_M3IF	Reserved
LLWU_M4IF	Reserved
LLWU_M5IF	Reserved
LLWU_M6IF	Reserved
LLWU_M7IF	Reserved

3.4.6 MMDVSQ 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.

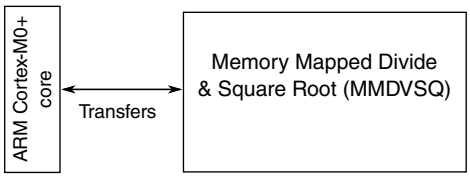


Figure 3-8. MMDVSQ configuration

Table 3-17. Reference links to related information

Topic	Related module	Reference
Full description	Memory Mapped Divide & Square root module(MMDVSQ)	MMDVSQ
System memory map		System memory map
Clocking		Clock distribution
Private Peripheral Bus (PPB)	ARM Cortex-M0+ core	ARM Cortex-M0+ core

3.4.7 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.

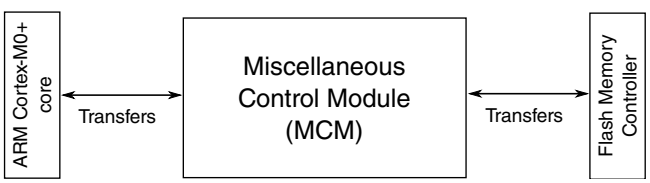


Figure 3-9. MCM configuration

Table 3-18. 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
Private Peripheral Bus (PPB)	ARM Cortex-M0+ core	ARM Cortex-M0+ core
Transfer	Flash memory controller	Flash memory controller

3.4.8 Crossbar-Light Switch 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.

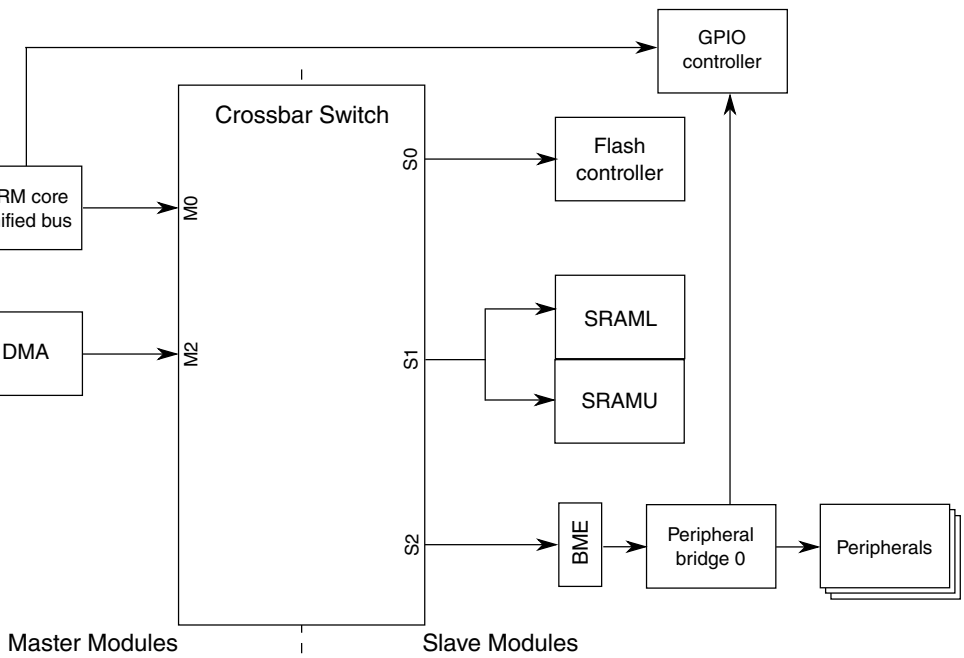


Figure 3-10. Crossbar-Light switch integration

Table 3-19. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	Crossbar Switch
System memory map		System memory map
Clocking		Clock Distribution
Crossbar switch master	ARM Cortex-M0+ core	ARM Cortex-M0+ core
Crossbar switch master	DMA controller	DMA controller
Crossbar switch slave	Flash memory controller	Flash memory controller
Crossbar switch slave	SRAM controller	SRAM configuration
Crossbar switch slave	Peripheral bridge	Peripheral bridge
2-ported peripheral	GPIO controller	GPIO controller

3.4.8.1 Crossbar-Light Switch Master Assignments

The masters connected to the crossbar switch are assigned as follows:

Master module	Master port number
ARM core unified bus	0
DMA	2

3.4.8.2 Crossbar Switch Slave Assignments

This device contains 3 slaves connected to the crossbar switch.

The slave assignment is as follows:

Slave module	Slave port number
Flash memory controller	0
SRAM controller	1
Peripheral bridge 0	2

3.4.9 Peripheral Bridge 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.

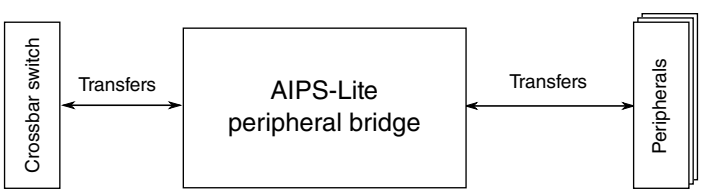


Figure 3-11. Peripheral bridge configuration

Table 3-20. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge (AIPS-Lite)	Peripheral bridge (AIPS-Lite)
System memory map		System memory map
Clocking		Clock Distribution
Crossbar switch	Crossbar switch	Crossbar switch

3.4.9.1 Number of peripheral bridges

This device contains one peripheral bridge.

3.4.9.2 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this device. See AIPS0 Memory Map for the memory slot assignment for each module.

3.4.10 DMA request multiplexer 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.

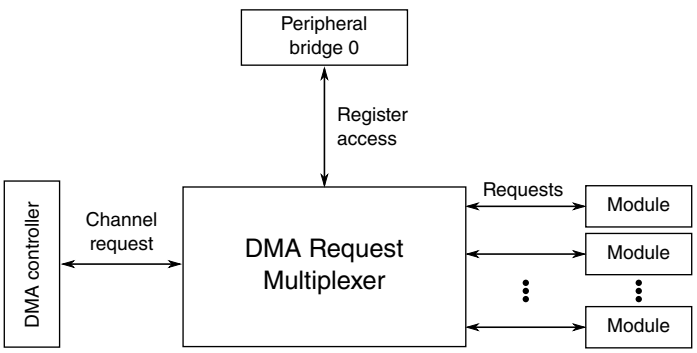


Figure 3-12. DMA request multiplexer configuration

Table 3-21. Reference links to related information

Topic	Related module	Reference
Full description	DMA request multiplexer	DMA Mux
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Channel request	DMA controller	DMA Controller
Requests		DMA request sources

3.4.10.1 Periodic trigger not supported

The periodic trigger capability is not supported with this device. Information related to this must be ignored.

3.4.10.2 DMA MUX Request Sources

This device includes a DMA request mux that allows up to 63 DMA request signals to be mapped to any of the 8 DMA channels. Because of the mux there is no hard correlation between any of the DMA request sources and a specific DMA channel. Some of the modules support Asynchronous DMA operation as indicated by the last column in the following DMA source assignment table.

Table 3-22. DMA request sources - MUX 0[1]

Source number	Source module	Source description	Async DMA capable
0	—	Channel disabled[2]	
1	Reserved	Not used	
2	UART0	Receive	
3	UART0	Transmit	
4	UART1	Receive	
5	UART1	Transmit	
6	Reserved	—	
7	Reserved	—	
8	Reserved	—	
9	Reserved	—	
10	Reserved	—	
11	Reserved	—	
12	Reserved	—	
13	Reserved	—	
14	FlexCAN0	Receive buffer full	
15	Reserved	—	
16	SPI0	Receive	
17	SPI0	Transmit	
18	Reserved	—	
19	Reserved	—	
20	Reserved	—	
21	Reserved	—	
22	I^2C0	Transmit/Receive/Arbitration lost/others	
23	Reserved	—	
24	FTM0	Channel 0	
25	FTM0	Channel 1	
26	FTM0	Channel 2	
27	FTM0	Channel 3	
28	FTM0	Channel 4	
29	FTM0	Channel 5	
30	FTM4	Channel 0	

Table continues on the next page...

Table 3-22. DMA request sources - MUX 0[1] (continued)

Source number	Source module	Source description	Async DMA capable
31	FTM4	Channel 1	
32	FTM1	Channel 0	
33	FTM1	Channel 1	
34	FTM2	Channel 0	
35	FTM2	Channel 1	
36	FTM3	Channel 0	
37	FTM3	Channel 1	
38	FTM3	Channel 2	
39	FTM3	Channel 3	
40	ADC0	Conversion complete	Yes
41	ADC1	Conversion complete	Yes
42	CMP0	—	Yes
43	CMP1	—	Yes
44	Reserved	—	
45	DAC0	Buffer pointer reaches upper or lower limit	
46	Reserved	—	
47	PDB1	—	
48	PDB0	—	
49	Port control module	Port A	Yes
50	Port control module	Port B	Yes
51	Port control module	Port C	Yes
52	Port control module	Port D	Yes
53	Port control module	Port E	Yes
54	FTM3	Channel 4	
55	FTM3	Channel 5	
56	FTM5	Channel 0	
57	FTM5	Channel 1	
58	DMA MUX	Always enabled	
59	DMA MUX	Always enabled	
60	DMA MUX	Always enabled	
61	DMA MUX	Always enabled	
62	DMA MUX	Always enabled	
63	DMA MUX	Always enabled	

1. If Async DMA capable field for a source is empty, it implies it is No.
2. Configuring a DMA channel to select source 0 or any of the reserved sources disables that DMA channel.

3.4.11 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.

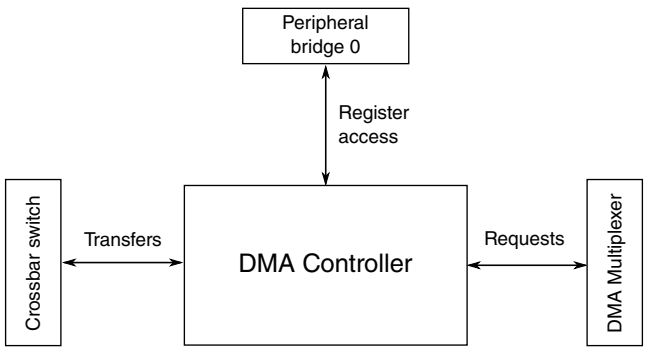


Figure 3-13. DMA Controller configuration

Table 3-23. 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
Crossbar switch	Crossbar switch	Crossbar switch
Requests		DMA request sources

3.4.12 DMA Channel output assignments

Each of the eight DMA channels provide a dma_done signal which determines that the data transfer has completed. These signals are provide as inputs to the PDB thus providing a mechanism to trigger another peripheral event.

Table 3-24. DMA channel ouput assignments

DMA channel	Description	Peripheral connection
Channel 0	dma0_done	PDB0 input 0100
Channel 1	dma1_done	PDB0 input 0101
Channel 2	dma2_done	PDB0 input 0110
Channel 3	dma3_done	PDB0 input 0111
Channel 4	dma4_done	PDB1 input 0100

Table continues on the next page...

Table 3-24. DMA channel ouput assignments (continued)

DMA channel	Description	Peripheral connection
Channel 5	dma5_done	PDB1 input 0101
Channel 6	dma6_done	PDB1 input 0110
Channel 7	dma7_done	PDB1 input 0111

3.4.13 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.

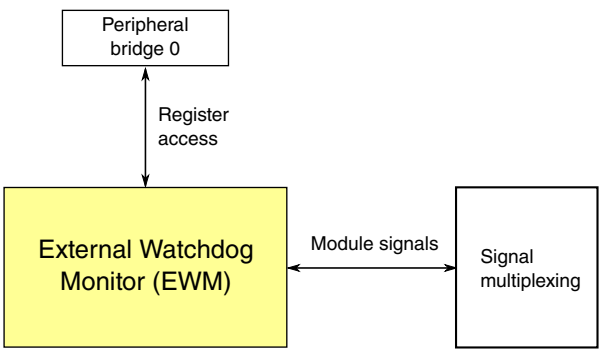


Figure 3-14. External Watchdog Monitor configuration

Table 3-25. 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	Port Control Module	Signal multiplexing

3.4.13.1 EWM clocks

This table shows the EWM clocks and the corresponding chip clocks.

This device has only one low-power clock, lpo_clk[0] which can be selected by writing 00 to EWM_CLKCTRL[CLKSEL] bit.

Table 3-26. EWM clock connections

Module clock	Chip clock
Low Power Clock	1 kHz LPO Clock

3.4.13.2 EWM low-power modes

The EWM module operates normally in Wait mode. This table shows the EWM low-power modes and the corresponding chip low-power modes.

Table 3-27. EWM low-power modes

Module mode	Chip mode
Wait	Wait, VLPW
Stop	Stop, VLPS

3.4.13.3 $\overline{\text{EWM\_OUT}}$ pin state in low power modes

When the CPU enters a Run mode from Wait or Stop recovery, the pin resumes its previous state before entering Wait or Stop mode. When the CPU enters Run mode from Power Down, the pin returns to its reset state.

3.4.14 Watchdog 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.

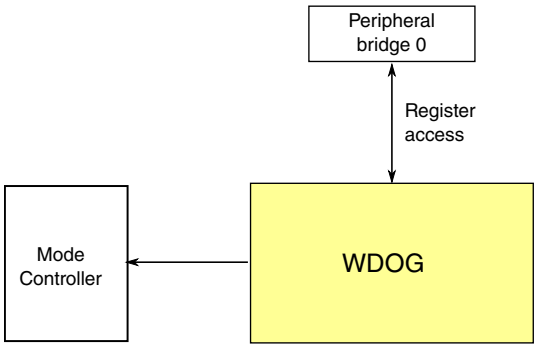


Figure 3-15. Watchdog configuration

Table 3-28. Reference links to related information

Topic	Related module	Reference
Full description	Watchdog	Watchdog
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
	Mode Controller (MC)	

3.4.14.1 WDOG clocks

This table shows the WDOG module clocks and the corresponding chip clocks.

Table 3-29. WDOG clock connections

Module clock	Chip clock
LPO Oscillator	1 kHz LPO Clock or MCGIRCLK (Depending on SIM_WDOGCTRL[WDGCLKS])
Alt Clock	Bus Clock
Fast Test Clock	Bus Clock
System Bus Clock	Bus Clock

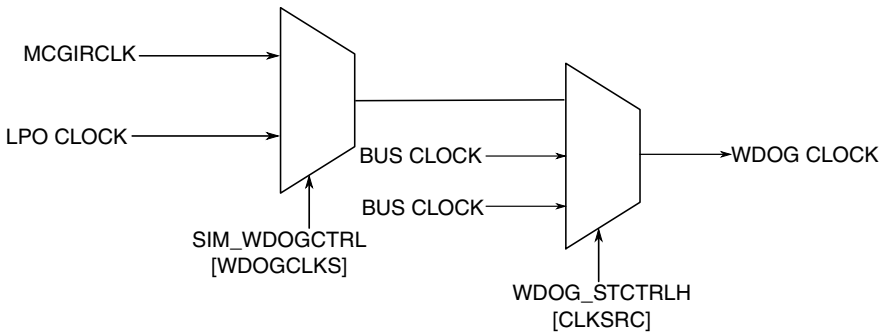


Figure 3-16. WDOG clock connections

3.4.14.2 WDOG low-power modes

This table shows the WDOG low-power modes and the corresponding chip low-power modes.

Table 3-30. WDOG low-power modes

Module mode	Chip mode
Wait	Wait, VLPW
Stop	Stop, VLPS
Power Down	VLLSx

3.5 Clock Modules

3.5.1 MCG 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.

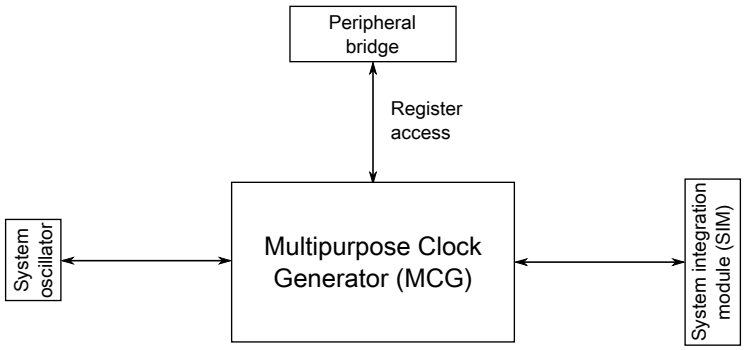


Figure 3-17. MCG configuration

Table 3-31. Reference links to related information

Topic	Related module	Reference
Full description	MCG	MCG
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

3.5.2 OSC 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.

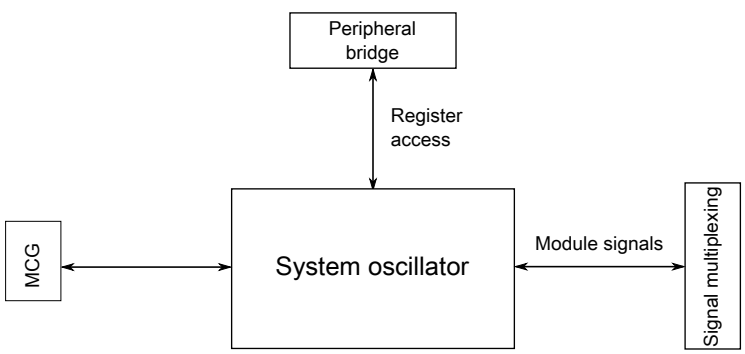


Figure 3-18. OSC configuration

Table 3-32. Reference links to related information

Topic	Related module	Reference
Full description	OSC	OSC
System memory map		System memory map
Clocking		Clock distribution

Table continues on the next page...

Table 3-32. Reference links to related information (continued)

Topic	Related module	Reference
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing
Full description	MCG	MCG

3.5.2.1 OSC modes of operation with MCG

The most common method of controlling the OSC block is through MCG clock source selection MCG_C1[CLKS] and the MCG_C2 register bits to configure for crystal or external clock operation. The OSC_CR also provides control for enabling the OSC and configuring internal load capacitors for the EXTAL and XTAL pins. See the OSC and MCG chapters for more details.

3.6 Memories and Memory Interfaces

3.6.1 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.

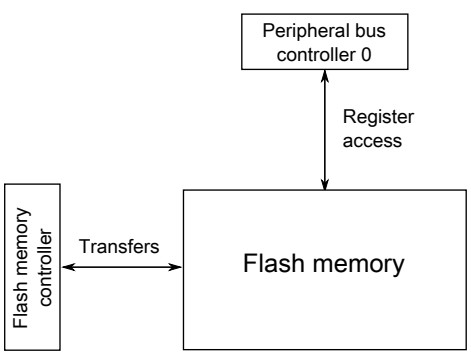


Figure 3-19. Flash memory configuration

Table 3-33. Reference links to related information

Topic	Related module	Reference
Full description	Flash memory	Flash memory
System memory map		System memory map
Clocking		Clock distribution

Table continues on the next page...

Table 3-33. Reference links to related information (continued)

Topic	Related module	Reference
Transfers	Flash memory controller	Flash memory controller
Register access	Peripheral bridge	Peripheral bridge

3.6.1.1 KV11x Flash Memory Sizes

The KV11 has 128 KB flash memory.

The flash access time supports two cycle reads at the maximum operating frequency of 25 MHz. The CPU can access the flash via the FMC (Flash Memory Controller) at 1:1 clock ratio up to 25 MHz CPU speed. Higher CPU frequency from 25 MHz to 75 MHz, the FMC will implement a 3:1 clock ratio.

Table 3-34. Flash memory size

Device	Program flash (KB)	Block 0 (P-Flash) address range
MKV11Z128VLH7	128	0x0000_0000 to 0x0001_FFFF
MKV11Z128VLF7	128	0x0000_0000 to 0x0001_FFFF
MKV11Z128VLC7	128	0x0000_0000 to 0x0001_FFFF
MKV11Z128VFM7	128	0x0000_0000 to 0x0001_FFFF
MKV11Z64VLH7	64	0x0000_0000 to 0x0000_FFFF
MKV11Z64VLF7	64	0x0000_0000 to 0x0000_FFFF
MKV11Z64VLC7	64	0x0000_0000 to 0x0000_FFFF
MKV11Z64VFM7	64	0x0000_0000 to 0x0000_FFFF
MKV10Z64VLH7	64	0x0000_0000 to 0x0000_FFFF
MKV10Z64VLF7	64	0x0000_0000 to 0x0000_FFFF
MKV10Z64VLC7	64	0x0000_0000 to 0x0000_FFFF
MKV10Z64VFM7	64	0x0000_0000 to 0x0000_FFFF
MKV10Z128VLH7	128	0x0000_0000 to 0x0001_FFFF
MKV10Z128VLF7	128	0x0000_0000 to 0x0001_FFFF
MKV10Z128VLC7	128	0x0000_0000 to 0x0001_FFFF
MKV10Z128VFM7	128	0x0000_0000 to 0x0001_FFFF
MKV11Z128VLH7P	120	0x0000_0000 to 0x0001DFFF
MKV11Z128VLC7P	120	0x0000_0000 to 0x0001DFFF
MKV11Z128VFM7P	120	0x0000_0000 to 0x0001DFFF
MKV10Z64VFM7P	56	0x0000_0000 to 0x0000_DFFF
MKV10Z64VLH7P	56	0x0000_0000 to 0x0000_DFFF
MKV10Z64VLF7P	56	0x0000_0000 to 0x0000_DFFF
MKV10Z64VLC7P	56	0x0000_0000 to 0x0000_DFFF

3.6.1.2 Flash Memory Map

The flash memory and the flash registers are located at different base addresses as shown in the following figure. The base address for each is specified in System memory map.

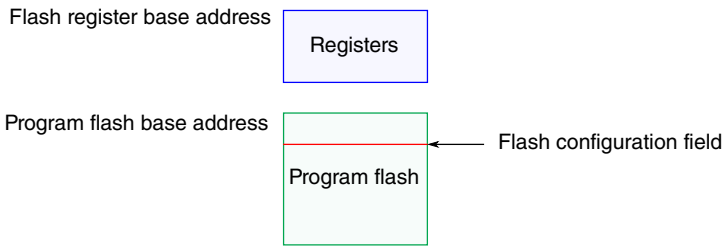


Figure 3-20. Flash memory map

The on-chip Flash is implemented in a portion of the allocated Flash range to form a contiguous block in the memory map beginning at address 0x0000_0000. See Flash Memory Sizes for details of supported ranges.

Accesses to the flash memory ranges outside the amount of Flash on the device causes the bus cycle to be terminated with an error followed by the appropriate response in the requesting bus master. Read collision events in which flash memory is accessed while a flash memory resource is being manipulated by a flash command also generates a bus error response.

3.6.1.3 Flash configuration and access control values

The following table describes the factory settings of the Flash Configuration field from address 0x0_0400 through 0x0_040F. The Flash is protected from mass erasure from the MDM-AP debug register or from an Erase All Blocks flash command. Since the values in the flash configuration are erased with the with a sector erase of sector 0 to re-establish these protection the FPROT, FSEC need to be re-programmed. See Flash Configuration Field Description for details on this section of memory.

Table 3-35. Flash configuration values for KMS enabled devices (cannot be changed)

Offset address	Value (if MCU is KMS)	Flash Size	Description
0x0_0400 - 0x0_0403	0xFF 0xFF 0xFF 0xFF		no value for back-door comparison key
0x0_0404 - 0x0_0407	0xFF 0xFF 0xFF 0xFF		
0x0_0408 - 0x0_040B	0xFF 0xFF 0xFF 0x0F	64 KB	top 8 KB protected
0x0_0408 - 0x0_040B	0xFF 0xFF 0xFF 0x3F	128 KB	

Table continues on the next page...

Table 3-35. Flash configuration values for KMS enabled devices (cannot be changed) (continued)

Offset address	Value (if MCU is KMS)	Flash Size	Description
0x0_040C	0xEE		not-secure, mass erase disabled
0x0_040D	0xFF		refer to FOPT
0x0_040E	0xFF		Reserved
0x0_040F	0xFF		Reserved

The following table describes the settings of the flash access controls. The flash access controls for a KMS enabled device cannot be erased or re-programmed. See Flash Access Protection for more information about these registers.

Table 3-36. Flash access control settings for KMS enabled devices (cannot be changed)

Offset address	Value (if MCU is KMS)	Flash Size	Description
0x4002_001C	0x0F	64 KB	XACCL0 top 8 KB read-only
	0x3F	128 KB	
0x4002_001D	0xFF		XACCL1
0x4002_001E	0xFF		XACCL2
0x4002_001F	0xFF		XACCL3

3.6.1.4 Flash Security

KMS enabled parts fully utilize the Flash Access Controls (FAC) security tool that protect a portion of the flash memory from being read. The KMS library is pre-programmed into the top 8 KB of flash. For these devices the FAC is not available for any other use. See Chip Security to know how flash security is implemented on this device .

3.6.1.5 Flash Modes

The flash memory chapter defines two modes of operation - NVM normal and NVM special modes.

3.6.1.6 Erase All Flash Contents

In addition to software, the entire flash memory may be erased external to the flash memory via the SW-DP debug port by setting MDM-AP CONTROL[0]. MDM-AP STATUS[0] is set to indicate the mass erase command has been accepted. MDM-AP STATUS[0] is cleared by power-on reset (POR).

Warning

For KMS enabled devices, do not use the MDM-AP Controls to mass erase the flash memory. Doing so will disable the KMS functionality.

3.6.1.7 FTFA_FOPT Register

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

3.6.2 Flash Memory 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.

See MCM_PLACR register description for details on the reset configuration of the FMC.

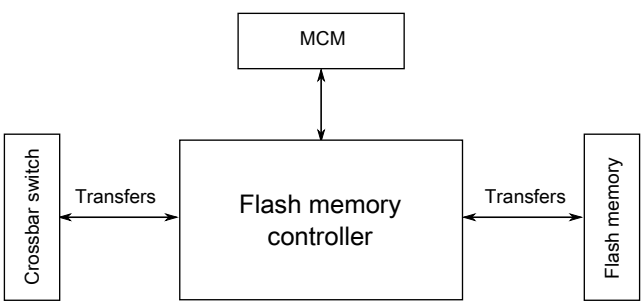


Figure 3-21. Flash memory controller configuration

Table 3-37. 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

Table continues on the next page...

Table 3-37. Reference links to related information (continued)

Topic	Related module	Reference
Transfers	Crossbar switch	Crossbar Switch
Register access	MCM	MCM

3.6.3 SRAM Configuration

This section summarizes how the module has been configured in the chip.

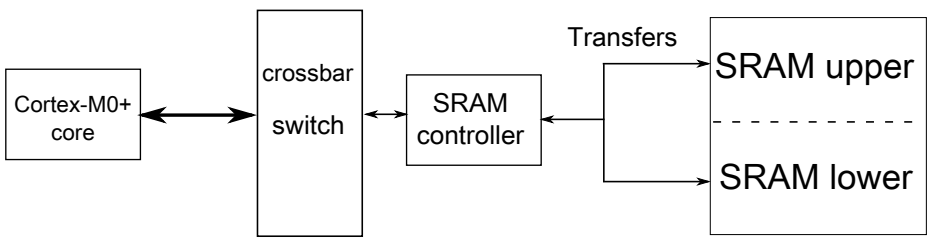


Figure 3-22. SRAM configuration

Table 3-38. Reference links to related information

Topic	Related module	Reference
Full description	SRAM	SRAM
System memory map		System memory map
Clocking		Clock Distribution
ARM Cortex-M0+ core		ARM Cortex-M0+ core

3.6.3.1 SRAM Sizes

This device contains SRAM which could be accessed by bus masters through the cross-bar switch. The amount of SRAM for the devices covered in this document is shown in the following table.

Table 3-39. SRAM memory size

Device	SRAM (KB)
MKV11Z128VLH7	16
MKV11Z128VLF7	16
MKV11Z128VLC7	16
MKV11Z128VFM7	16

Table continues on the next page...

Table 3-39. SRAM memory size (continued)

Device	SRAM (KB)
MKV11Z64VLH7	16
MKV11Z64VLF7	16
MKV11Z64VLC7	16
MKV11Z64VFM7	16
MKV10Z64VLH7	16
MKV10Z64VLF7	16
MKV10Z64VLF7	16
MKV10Z64VFM7	16
MKV10Z128VLH7	16
MKV10Z128VLF7	16
MKV10Z128VLC7	16
MKV10Z128VFM7	16

3.6.3.2 SRAM Ranges

The on-chip SRAM is split into two ranges, 1/4 is allocated SRAM_L and 3/4 is allocated to SRAM_U.

The on-chip RAM is implemented such that the SRAM_L and SRAM_U ranges form a contiguous block in the memory map. As such:

- SRAM_L is anchored to 0x1FFF_FFFF and occupies the space before this ending address.
- SRAM_U is anchored to 0x2000_0000 and occupies the space after this beginning address.

Valid address ranges for SRAM_L and SRAM_U are then defined as:

- SRAM_L = [0x2000_0000–(SRAM_size/4)] to 0x1FFF_FFFF
- SRAM_U = 0x2000_0000 to [0x2000_0000+(SRAM_size*(3/4))-1]

This is illustrated in the following figure.

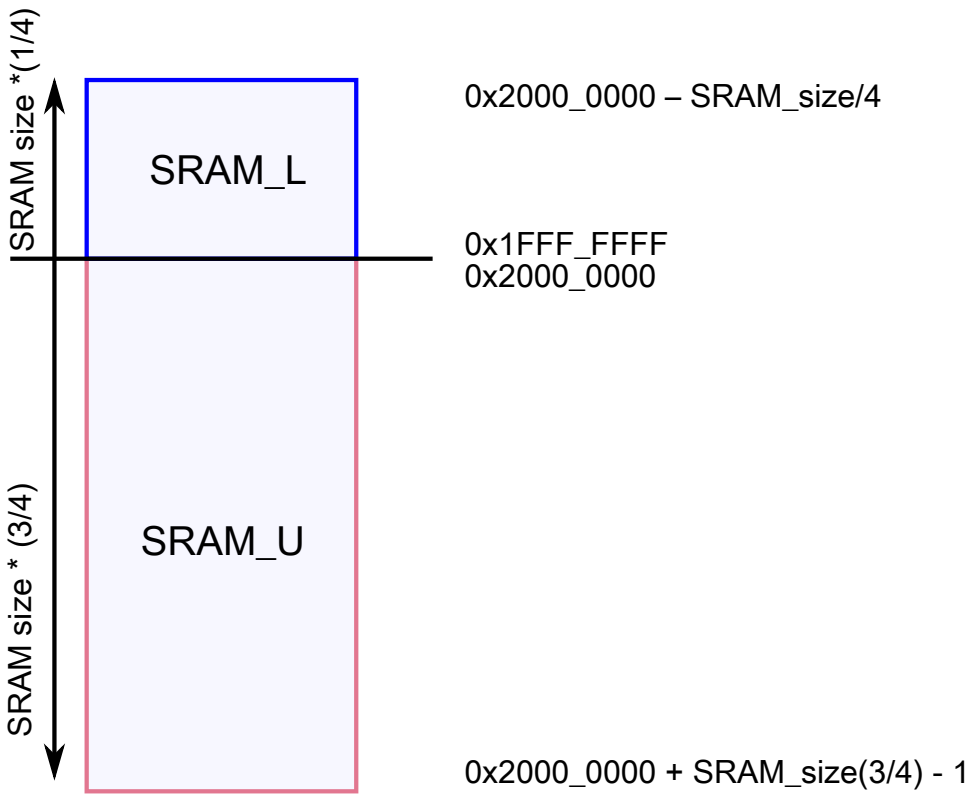


Figure 3-23. SRAM blocks memory map

For example, for a device containing 16 KB of SRAM the ranges are:

- SRAM_L: 0x1FFF_F000 – 0x1FFF_FFFF
- SRAM_U: 0x2000_0000 – 0x2000_2FFF

3.6.3.3 SRAM retention in low power modes

The SRAM is retained down to VLLS3 mode. In VLLS1 and VLLS0 no SRAM is retained.

3.7 Security

3.7.1 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.

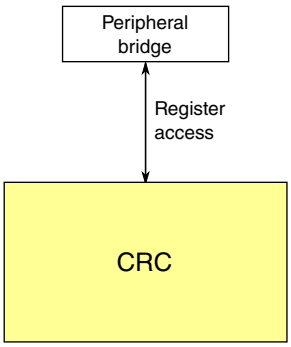


Figure 3-24. CRC configuration

Table 3-40. Reference links to related information

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

3.8 Analog

3.8.1 16-bit SAR 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.

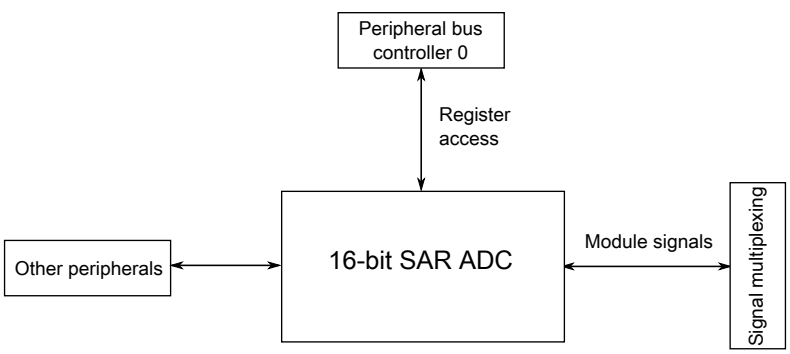


Figure 3-25. 16-bit SAR ADC configuration

Table 3-41. Reference links to related information

Topic	Related module	Reference
Full description	16-bit SAR ADC	16-bit SAR ADC
System memory map		System memory map

Table continues on the next page...

Table 3-41. Reference links to related information (continued)

Topic	Related module	Reference
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

3.8.1.1 Kv11x ADC Instantiation Information

This device contains two 16-bit successive approximation ADCs, ADC0 and ADC1. ADCA has 15 channels. ADCB has 18 channels.

Each ADC supports both software and hardware triggers. The hardware trigger sources are listed in the Module-to-Module section.

The targeted number of ADC channels present on the device is determined by the pinout of the specific device package and is shown in the following table. On completion of pinout this will be concluded.

3.8.1.2 DMA Support on ADC

Applications may require continuous sampling of the ADC that may have considerable load on the CPU. The ADC supports DMA request functionality for higher performance when the ADC is sampled at a very high rate. The ADC can trigger the DMA (via DMA req) on conversion completion.

3.8.1.3 ADC0 Connections/Channel Assignment

NOTE

As indicated by the following sections, each ADCx_DPx input and certain ADCx_DMx inputs may operate as single-ended ADC channels in single-ended mode.

3.8.1.3.1 ADC0 Channel Assignment

ADC Channel (SC1n[ADCH])	Channel	Input signal (SC1n[DIFF]= 1)	Input signal (SC1n[DIFF]= 0)
00000	AD0	ADC0_DP0, ADC0_DM0	ADC0_SE0
00001	AD1	ADC0_DP1, ADC0_DM1	ADC0_SE1

Table continues on the next page...

ADC Channel (SC1n[ADCH])	Channel	Input signal (SC1n[DIFF]= 1)	Input signal (SC1n[DIFF]= 0)
00010	AD2	Reserved	ADC0_SE2
00011	AD3	Reserved	ADC0_SE3
00100	AD4	Reserved	ADC0_SE4
00101	AD5	Reserved	ADC0_SE5
00110	AD6	Reserved	ADC0_SE6
00111	AD7	Reserved	ADC0_SE7
01000	AD8	Reserved	ADC0_SE8
01001	AD9	Reserved	ADC0_SE9
01011	AD10	Reserved	ADC0_SE10
01100	AD11	Reserved	ADC0_SE11
01101	AD12	Reserved	ADC0_SE12
01110	AD13	Reserved	ADC0_SE13
01111	AD14	Reserved	ADC0_SE14
10000	Reserved	Reserved	Reserved
10001	Reserved	Reserved	Reserved
10010	Reserved	Reserved	Reserved
10011	Reserved	Reserved	Reserved
10100	Reserved	Reserved	Reserved
10101	Reserved	Reserved	Reserved
10110	Reserved	Reserved	Reserved
10111	AD23	Reserved	Reserved
11000	Reserved	Reserved	Reserved
11001	AD25	-VREFH (Diff)	VREFH
11010	AD26	Temperature Sensor (Diff)	Temperature Sensor (S.E)
11011	AD27	Bandgap (Diff)	Bandgap (S.E)[1]
11100	AD28	Reserved	Reserved
11101	AD29	Reserved	Reserved
11110	AD30	Reserved	Reserved
11111	AD31	Module Disabled	Module Disabled

1. This is the PMC bandgap 1V reference voltage. Prior to reading from this ADC channel, ensure that you enable the bandgap buffer by setting the PMC_REGSC[BGBE] bit. Refer to the device data sheet for the bandgap voltage (V_{BG}) specification.

3.8.1.4 ADC1 Connections/Channel Assignment

NOTE

As indicated by the following sections, each ADCx_DPx input and certain ADCx_DMx inputs may operate as single-ended ADC channels in single-ended mode.

3.8.1.4.1 ADC1 Channel Assignment

ADC Channel (SC1n[ADCH])	Channel	Input signal (SC1n[DIFF]= 1)	Input signal (SC1n[DIFF]= 0)
00000	AD0	Reserved	ADC1_SE0
00001	AD1	ADC1_DP1, ADC1_DM1	ADC1_SE1
00010	AD2	ADC1_DP2, ADC1_DM2	ADC1_SE2
00011	AD3	Reserved	ADC1_SE3
00100	AD4	Reserved	ADC1_SE4
00101	AD5	Reserved	ADC1_SE5
00110	AD6	Reserved	ADC1_SE6
00111	AD7	Reserved	ADC1_SE7
01000	AD8	Reserved	ADC1_SE8
01001	AD9	Reserved	ADC1_SE9
01010	AD10	Reserved	ADC1_SE10
01011	AD11	Reserved	ADC1_SE11
01100	AD12	Reserved	ADC1_SE12
01101	AD12	Reserved	ADC1_SE13
01110	AD13	Reserved	ADC1_SE14
01111	AD14	Reserved	ADC1_SE15
10000	AD15	Reserved	ADC1_SE16
10001	AD16	Reserved	ADC1_SE17
10010	AD17	Reserved	Reserved
10011	Reserved	Reserved	Reserved
10100	Reserved	Reserved	Reserved
10101	Reserved	Reserved	Reserved
10110	Reserved	Reserved	Reserved
10111	AD23	Reserved	Reserved
11000	Reserved	Reserved	Reserved
11001	AD25	-VREFH (Diff)	VREFH
11010	AD26	Temperature Sensor (Diff)	Temperature Sensor (S.E)
11011	AD27	Bandgap (Diff)	Bandgap (S.E)[1]
11100	AD28	Reserved	Reserved
11101	AD29	Reserved	Reserved
11110	AD30	Reserved	Reserved
11111	AD31	Module Disabled	Module Disabled

1. This is the PMC bandgap 1V reference voltage. Prior to reading from this ADC channel, ensure that you enable the bandgap buffer by setting the PMC_REGSC[BGBE] bit. Refer to the device data sheet for the bandgap voltage (V_{BG}) specification.

3.8.1.5 ADC Analog Supply and Reference Connections

3.8.1.6 ADC Reference Options

The ADC supports the following references:

- VREFH/VREFL - connected as the primary reference option
- VDDA - connected as the V_{ALT} reference option

3.8.1.7 Alternate clock

For this device, the user can select the output of the third clock divider as the alternate clock source of the ADC. The user can also select OSCERCLK or IRCLK. For more details, see SIM_SOPT7[ADC0ALTCLKSRC] and SIM_SOPT7[ADC1ALTCLKSRC].

NOTE

This clock option is only usable when the clock output frequency is less than 24 MHz.

3.8.2 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.

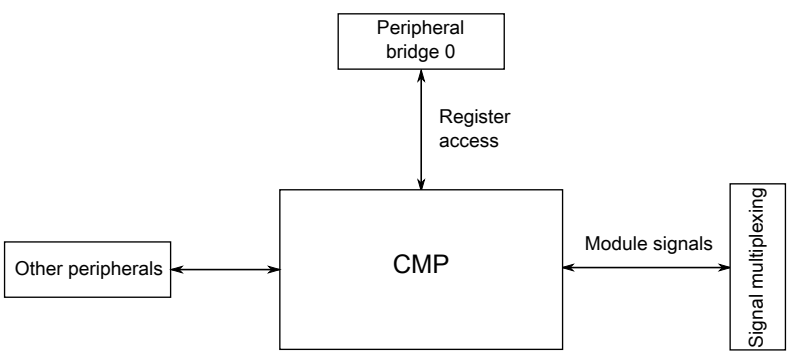


Figure 3-26. CMP configuration

Table 3-42. 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	Port control	Signal multiplexing

3.8.2.1 CMP instantiation information

The device includes two high speed comparator and two 8-input multiplexors for both the inverting and non-inverting inputs of the comparator. Each CMP input channel connects to both muxes. Two of the channels are connected to internal sources, leaving resources to support up to 6 input pins. See the channel assignment table for a summary of CMP input connections for this device.

The CMP also includes one 6-bit DAC with a 64-tap resistor ladder network, which provides a selectable voltage reference for applications where voltage reference is needed for internal connection to the CMP.

Also instantiated is a feed from the 12-bit DAC output to each of the CMPs.

Both the CMPs can be optionally on in all modes except VLLS0.

The CMP has several module to module interconnects in order to facilitate ADC triggering, FTM triggering, and UART IR interfaces. For complete details on the CMP module interconnects please refer to the Module-to-Module section.

3.8.2.2 CMP input connections

The following table shows the fixed internal connections to the CMP.

Table 3-43. CMP0 input connections

CMP Inputs	CMP0
IN0	CMP0_IN0
IN1	CMP0_IN1
IN2	CMP0_IN2
IN3	CMP0_IN3
IN4	CMP0_IN4
IN5	CMP0_IN5
IN6	Bandgap[1]
IN7	6-bit DAC0 reference

1. This is the PMC bandgap 1V reference voltage. Prior to using as CMP input, ensure that you enable the bandgap buffer by setting the PMC_REGSC[BGBE] bit. Refer to the device data sheet for the bandgap voltage (V_{BG}) specification.

Table 3-44. CMP1 input connections

CMP Inputs	CMP1
IN0	CMP1_IN0

Table continues on the next page...

Table 3-44. CMP1 input connections (continued)

CMP Inputs	CMP1
IN1	CMP1_IN1
IN2	Reserved
IN3	Reserved
IN4	CMP1_IN4
IN5	CMP1_IN5
IN6	Bandgap[1]
IN7	6-bit DAC0 reference

1. This is the PMC bandgap 1V reference voltage. Prior to using as CMP input, ensure that you enable the bandgap buffer by setting the PMC_REGSC[BGBE] bit. Refer to the device data sheet for the bandgap voltage (V_{BG}) specification.

3.8.2.3 CMP external references

The 6-bit DAC sub-block supports selection of two references. For this device, the references are connected as follows:

- VREFH - V_{in1} input. When using VREFH, any ADC conversion using this same reference at the same time is negatively impacted.
- VDD - V_{in2} input

3.8.2.4 CMP trigger mode

The CMP and 6-bit DAC sub-block supports trigger mode operation when the CMP_CR1[TRIGM] is set. When trigger mode is enabled, the trigger event will initiate a compare sequence that must first enable the CMP and DAC prior to performing a CMP operation and capturing the output. In this device, control for this two staged sequencing is provided from the LPTMR. The LPTMR triggering output is always enabled when the LPTMR is enabled. The first signal is supplied to enable the CMP and DAC and is asserted at the same time as the TCF flag is set. The delay to the second signal that triggers the CMP to capture the result of the compare operation is dependent on the LPTMR configuration. In Time Counter mode with prescaler enabled, the delay is 1/2 Prescaler output period. In Time Counter mode with prescaler bypassed, the delay is 1/2 Prescaler clock period.

The delay between the first signal from LPTMR and the second signal from LPTMR must be greater than the Analog comparator initialization delay as defined in the device datasheet.

3.8.3 12-bit 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.

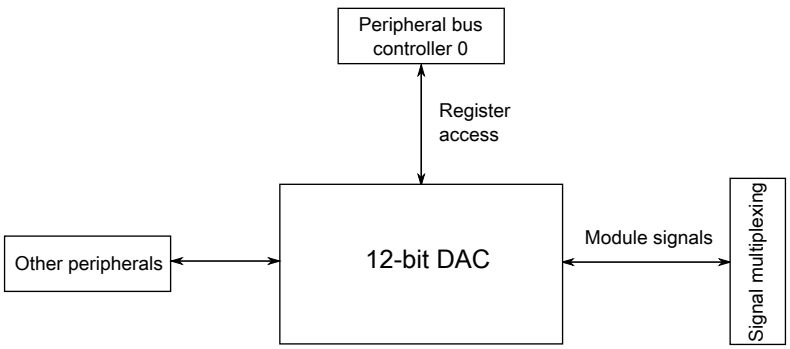


Figure 3-27. 12-bit DAC configuration

Table 3-45. 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	Port control	Signal multiplexing

3.8.3.1 12-bit DAC Instantiation Information

This device contains one 12-bit digital-to-analog converter (DAC) with programmable reference generator output. The DAC includes a two word FIFO for DMA support.

3.8.3.2 12-bit DAC Output

The output of the DAC can be placed on an external pin or selected as an input to the analog comparator or ADC.

3.8.3.3 12-bit DAC Analog Supply and Reference Connections

This device has separate VDDA from digital VDD and separate VSSA from VSS. VDDA and VSSA provide power to ADC0, ADC1, and DAC0.

This device contains separate VREFH and VREFL pins. On 32-pin packages, these are connected to VDDA and VSSA respectively using wire bonding. In 48-pin and 64-pin packages, these are separate from VDDA and VSSA.

3.8.3.4 12-bit DAC Reference

For this device VREFH and VDDA are selectable as the DAC reference. VREFH is connected to the DACREF_1 input and VDDA is connected to the DACREF_2 input. Use DACx_C0[DACRFS] control bit to select between these two options.

Be aware that if the DAC and ADC use the same reference simultaneously, some degradation of ADC accuracy is to be expected due to DAC switching.

3.9 Timers

3.9.1 FlexTimer 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.

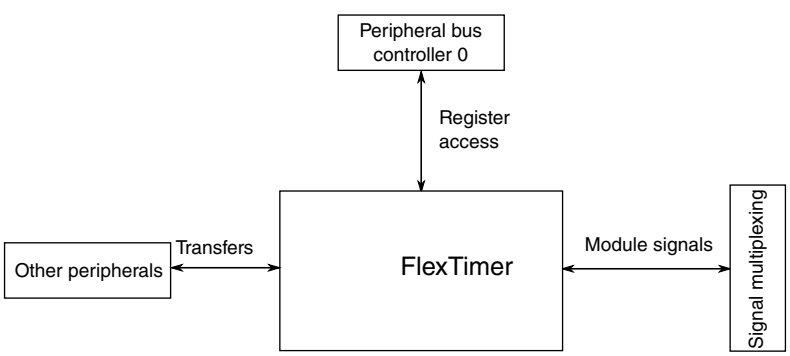


Figure 3-28. FTM configuration

Table 3-46. Reference links to related information

Topic	Related module	Reference
Full description	FlexTimer Module	FlexTimer Module
System memory map		System memory map

Table continues on the next page...

Table 3-46. Reference links to related information (continued)

Topic	Related module	Reference
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

3.9.1.1 Instantiation Information

This device contains six FlexTimer modules.

The following table shows how these modules are configured.

Table 3-47. FTM Instantiations

FTM instance	Number of channels	Features/usage
FTM0	6	3-phase motor
FTM1	2	Quadrature decoder or general purpose
FTM2	2	Quadrature decoder or general purpose
FTM3	6	3-phase motor
FTM4	2	Quadrature decoder or general purpose
FTM5	2	Quadrature decoder or general purpose

NOTE

FTM1, FTM2, and FTM5 have the PHA and PHB signals independently muxed out from the CH0 and CH1 signals. This allows users to implement quadrature decoding via the FTMx_QD_PHA/B pins, and in parallel can deploy input capture functions using the FTMx_CH0/1 pins. FTM4 has the PHA directly connected to its CH0 signal, PHB connected to its CH1 signal, and thus can perform either a quadrature decode or an input capture function, but not simultaneously.

All six FTMs can be synchronized with each other or run independently. FTM1 and FTM2 will be enhanced to support Hall Sensor detect and chopper mode and possibly blanking mode.

Each of the six FTMs, can be synchronized with each other via FTMxSYNCBITs in the SIM_SOPT8 register. For example, FTM0 and FTM4 can be synchronized together to appear in functionality as an 8-ch FTM, and FTM3 and FTM5 can by synchronized together to form another 8-ch FTM.

3.9.1.2 External Clock Options

By default each FTM is clocked by the system clock (the FTM refers to it as system clock). Each module contains a register setting that allows the module to be clocked from an external clock instead. There are three external FTM_CLKINx pins that can be selected by any FTM module via the SOPT4 register in the SIM module (SIM_SOPT4). Each of the 6 FTMs can be clocked by an independent external clock if required.

3.9.1.3 Fixed frequency clock

The fixed frequency clock for each FTM is MCGFFCLK. The MCGFFCLK clock source can be selected via the FTMFFCLKSEL bits in SIM_SOPT2 register.

3.9.1.4 FTM Interrupts

The FlexTimer has multiple sources of interrupt. However, these sources are OR'd together to generate a single interrupt request per FTM module to the interrupt controller. When an FTM interrupt occurs, read the FTM status registers (FTMx_FMS, FTMx_SC, and FTMx_STATUS) to determine the exact interrupt source.

3.9.1.5 FTM Fault Detection Inputs

The following fault detection input options for the FTM modules are selected via the SIM_SOPT4 and SIM_SOPT6 registers in the SIM module. The external pin option is selected by default.

- FTM0 FAULT0 = FTM0_FLT0 pin or CMP0_OUT
- FTM0 FAULT1 = FTM0_FLT1 pin or CMP1_OUT
- FTM0 FAULT2 = FTM0_FLT2 pin
- FTM0 FAULT3 = FTM0_FLT3 pin

- FTM1 FAULT0 = FTM1_FLT0 pin or CMP0_OUT
- FTM1 FAULT1 = CMP1_OUT

- FTM2 FAULT0 = FTM2_FLT0 pin or CMP0_OUT
- FTM2 FAULT1 = CMP1_OUT

- FTM3 FAULT0 = FTM3_FLT0 or CMP0_OUT
- FTM3 FAULT1 = CMP1_OUT

- FTM4 FAULT0 = FTM4_FLT0 pin or CMP0_OUT
- FTM4 FAULT1 = CMP1_OUT

- FTM5 FAULT0 = FTM5_FLT0 pin or CMP0_OUT
- FTM5 FAULT1 = CMP1_OUT

3.9.1.6 FTM Hardware Triggers

The FTM synchronization hardware triggers are connected in the chip as follows:

- FTM0 hardware trigger 0 - TRIG0 = SIM_SOPT8[FTM0SYNCBIT]/CMP0_OUT or FTM1_match
- FTM0 hardware trigger 1 - TRIG1 = PDB0 channel 1 trigger out or FTM2_match
- FTM0 hardware trigger 2 - TRIG2 = CMP0_OUT or CMP1_OUT (selectable via SIM control bit)

- FTM1 hardware trigger 0 - TRIG0 = SIM_SOPT8[FTM1SYNCBIT]/CMP0_OUT or FTM0_match
- FTM1 hardware trigger 1 - TRIG1 = PDB0 channel 1 trigger out or FTM2_match
- FTM1 hardware trigger 2 - TRIG2 = CMP0_OUT or CMP1_OUT (selectable via SIM control bit)

- FTM2 hardware trigger 0 - TRIG0 = SIM_SOPT8[FTM2SYNCBIT]/CMP0_OUT or FTM0_match
- FTM2 hardware trigger 1 - TRIG1 = PDB0 channel 1 trigger out or FTM1_match
- FTM2 hardware trigger 2 - TRIG 2 = CMP0_OUT or CMP1_OUT (selectable via SIM control bit)

- FTM3 hardware trigger 0 - TRIG0 = SIM_SOPT8[FTM3SYNCBIT]/CMP0_OUT or FTM5_match
- FTM3 hardware trigger 1 - TRIG1 = PDB1 channel 1 trigger out or FTM4_match
- FTM3 hardware trigger 2 - TRIG2 = CMP0_OUT or CMP1_OUT (selectable via SIM control bit)

- FTM4 hardware trigger 0 - TRIG0 = SIM_SOPT8[FTM4SYNCBIT]/CMP0_OUT or FTM3_match
- FTM4 hardware trigger 1 - TRIG1 = PDB1 channel 1 trigger out or FTM5_match
- FTM4 hardware trigger 2 - TRIG2 = CMP0_OUT or CMP1_OUT (selectable via SIM control bit)

- FTM5 hardware trigger 0 - TRIG0 = SIM_SOPT8[FTM5SYNCBIT]/CMP0_OUT or FTM3_match

- FTM5 hardware trigger 1 - TRIG1 = PDB1 channel 1 trigger out or FTM4_match
- FTM5 hardware trigger 2 - TRIG2 = CMP0_OUT or CMP1_OUT (selectable via SIM control bit)

For the triggers with more than one option, SIM_SOPT4 and SIM_SOPT6 registers control the selection.

FTM0SYNCBIT, FTM1SYNCBIT, FTM2SYNCBIT, FTM3SYNCBIT, FTM4SYNCBIT, and FTM5SYNCBIT fields are in the same SIM_SOPT8 register to allow the user to synchronize all six timers.

3.9.1.7 Input capture options for FTM module instances

The following channel 0 input capture source options for FTM1, and FTM2 are selected via the SIM_SOPT4 register. The external pin option is selected by default.

- FTM1 channel 0 input capture = FTM1_CH0 pin or CMP0 output or CMP1 output
- FTM2 channel 0 input capture = FTM2_CH0 pin or CMP0 output or CMP1 output

The following channel 0 input capture source options for FTM4 and FTM5 are selected via the SIM_SOPT6 register. The external pin option is selected by default.

- FTM4 channel 0 input capture = FTM4_CH0 pin or CMP0 output or CMP1 output
- FTM5 channel 0 input capture = FTM5_CH0 pin or CMP0 output or CMP1 output

3.9.1.8 FTM output triggers for other modules

FTM output triggers can be selected as input triggers for the PDB, which then can be used to provide triggers to other peripherals for example the ADC modules.

FTM0 all 6 channels when configured as output compare or PWM mode, will provide an output signal on a compare event. Each of the 6 channels output compare signals are OR'd together and this composite signal is used to trigger other peripheral actions via the PDBs via PDB0 ch 1000 and PDB1 ch 1000

FTM3 all 6 channels when configured as output compare or PWM mode, will provide an output signal on a compare event. Each of the 6 channels output compare signals are OR'd together and this composite signal is used to trigger other peripheral actions via the PDBs via PDB0 ch 1011 and PDB1 ch 1011.

Similarly FTM1's two output channels are OR'd together and the composite signal is used to trigger other peripherals via PDB0 ch 1001 and PDB1 ch 1001.

FTM2's two output channels are OR'd together and the composite signal is used to trigger other peripherals via PDB0 ch 1010 and PDB1 ch 1010.

FTM4's two output channels are OR'd together and the composite signal is used to trigger other peripherals via PDB0 ch 1100 and PDB1 ch 1100.

FTM5's two output channels are OR'd together and the composite signal is used to trigger other peripherals via PDB0 ch 1101 and PDB1 ch 1101.

FTM1's channel 0 output is also applied as an option to both UART0 and UART1 TXD input signal. This option is selected via the SIMx register and the FTM1 channel 0 signal is modulated with the corresponding TXD input pin signal.

FTM0 - Channel 0,1,2,3,4,5 and CNTINIT output to PDB input channel 1000

FTM1 - Channel 0,1 and CNTINIT output to PDB input channel 1001

FTM1 - Channel 0 output also applied to UART0 and UART1 TXD input (modulated with TXD pin)

FTM2 - Channel 0,1 and CNTINIT output to PDB input channel 1010

FTM3 - Channel 0,1,2,3,4,5 and CNTINIT output to PDB input channel 1011

FTM4 - Channel 0,1 and CNTINIT output to PDB input channel 1100

FTM5 - Channel 0,1 and CNTINIT output to PDB input channel 1101

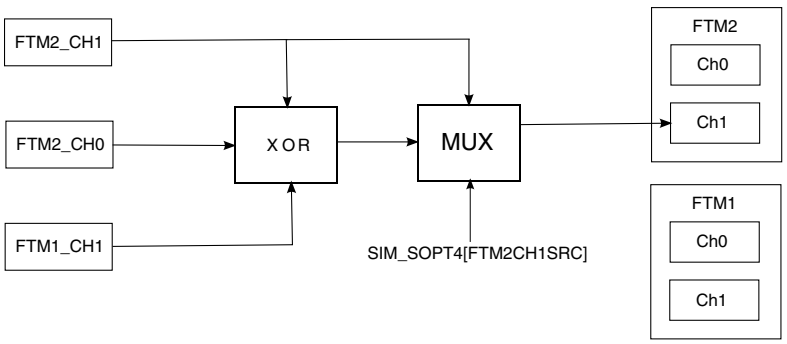
3.9.1.9 FTM Hall sensor support

For 3 phase motor control sensor-ed applications the use of Hall sensors, generally 3 sensors placed 120 degrees apart around the rotor, are deployed to detect position and speed. Each of the 3 sensors provides a pulse that applied to an input capture pin, can then be analyzed and both speed and position can be deduced.

This device has two 2-channel FTMs: FTM1 and FTM2, and thus provides 4 input capture pins. To simplify the calculations required by the CPU on each hall sensor's input, if all 3 inputs are "exclusively OR'd " into one timer channel and the free running counter is refreshed on every edge then this can simplify the speed calculation, removing the need to perform subtractions of previous edge captures with subsequent edge captures.

Thus, via the SIM module and SIM_SOPT4 register, the FTM2CH1SRC bit provides the choice of normal FTM2_CH1 input or the XOR of FTM2_CH0, FTM2_CH1 and FTM1_CH1 pins that will be applied to FTM2_CH1.

Note: If the user utilizes FTM1_CH1 to be an input to FTM2_CH1, FTM1_CH0 can still be utilized for other functions.



3.9.1.10 FTM modulation implementation

The FTM0, FTM2, FTM3, and FTM4 can support a modulation function where the output channels when configured as PWM or Output Compare mode will modulate another timer output when the channel signal is asserted. This function is supported via the SIM module.

Any of the 6 channels of FTM0 and FTM3 and any of the two channels of FTM2 and FTM4 can support this modulation function.

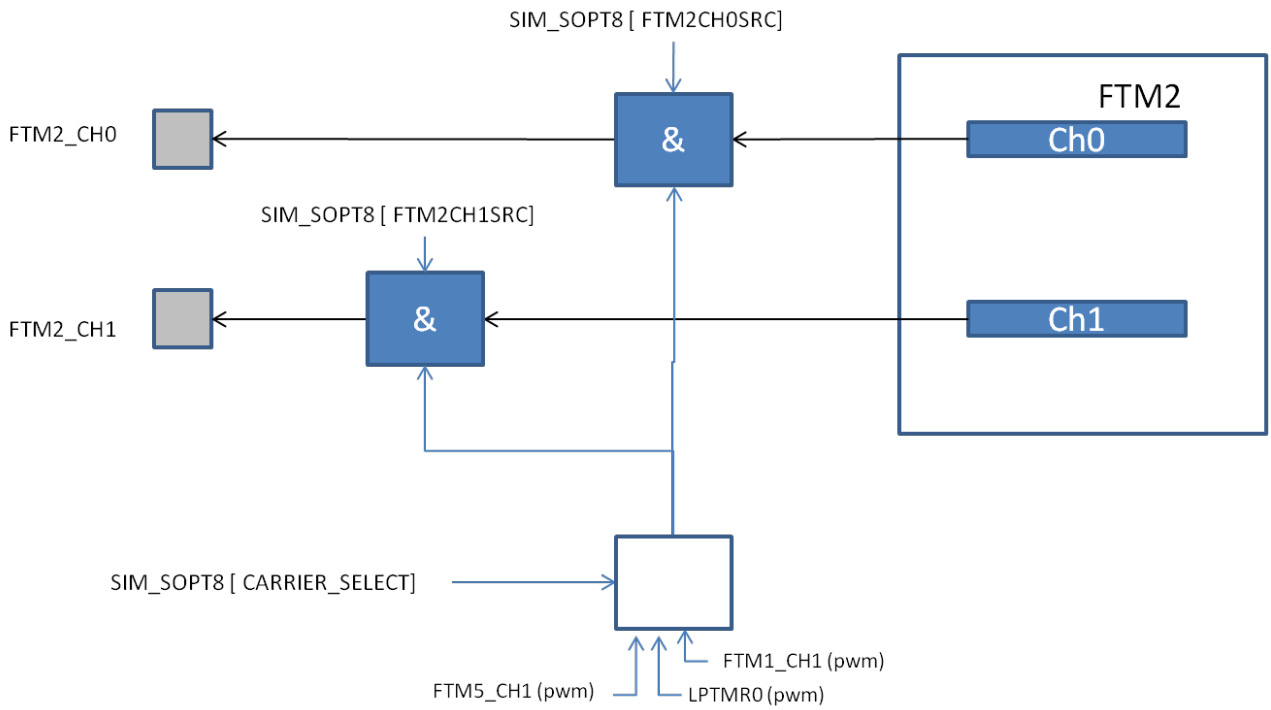
The SIM_SOPT8 and SIM_SOPT9 registers have two control bits to select one of three input carrier signals to be modulated. The SIM_SOPT8[CARRIER_SELECT0] and SIM_SOPT9[CARRIER_SELECT1] bits will select either the output of FTM1_CH1, FTM5_CH1 or the output of the LPTMR0 Prescaler.

Note: If FTM1_CH1 is used as the carrier input signal to be modulated, then the user must configure FTM1_CH1 to provide a signal that has a higher frequency of the FTM0/3 or FTM2/4 channel output. Also it limits the use of the FTM1_CH0 function, as the FTM1_CH1 will be programmed to provide a 50% duty PWM signal and limit the start and modulus values for the free running counter.

Similarly, if FTM5_CH1 is used as the carrier input signal to be modulated, then the user must configure FTM5_CH1 to provide a signal that has a higher frequency of the FTM0/3 or FTM2/4 channel output. Also it limits the use of the FTM5_CH0 function, as the FTM5_CH1 will be programmed to provide a 50% duty PWM signal and limit the start and modulus values for the free running counter.

The SIM_SOPT8 and SIM_SOPT9 registers have eight control bits FTMxCHySRC bits that allow the user to select normal PWM/Output Compare mode on the corresponding FTM timer channel or modulate the selected carrier signal on when the channel is

asserted. The diagram below shows the implementation for FTM2. FTM0 has similar implementation on each of its 6 channels.(like wise FTM3 and FTM4) See SIM Block Guide for further information.



FTMx modulation function

3.9.1.11 FTM Global Time Base

This chip provides the optional FTM global time base feature (see Global time base (GTB)).

FTM0 and FTM3 provide the source for the FTM global time base. The other FTM modules can share the time base as shown in the following figure:

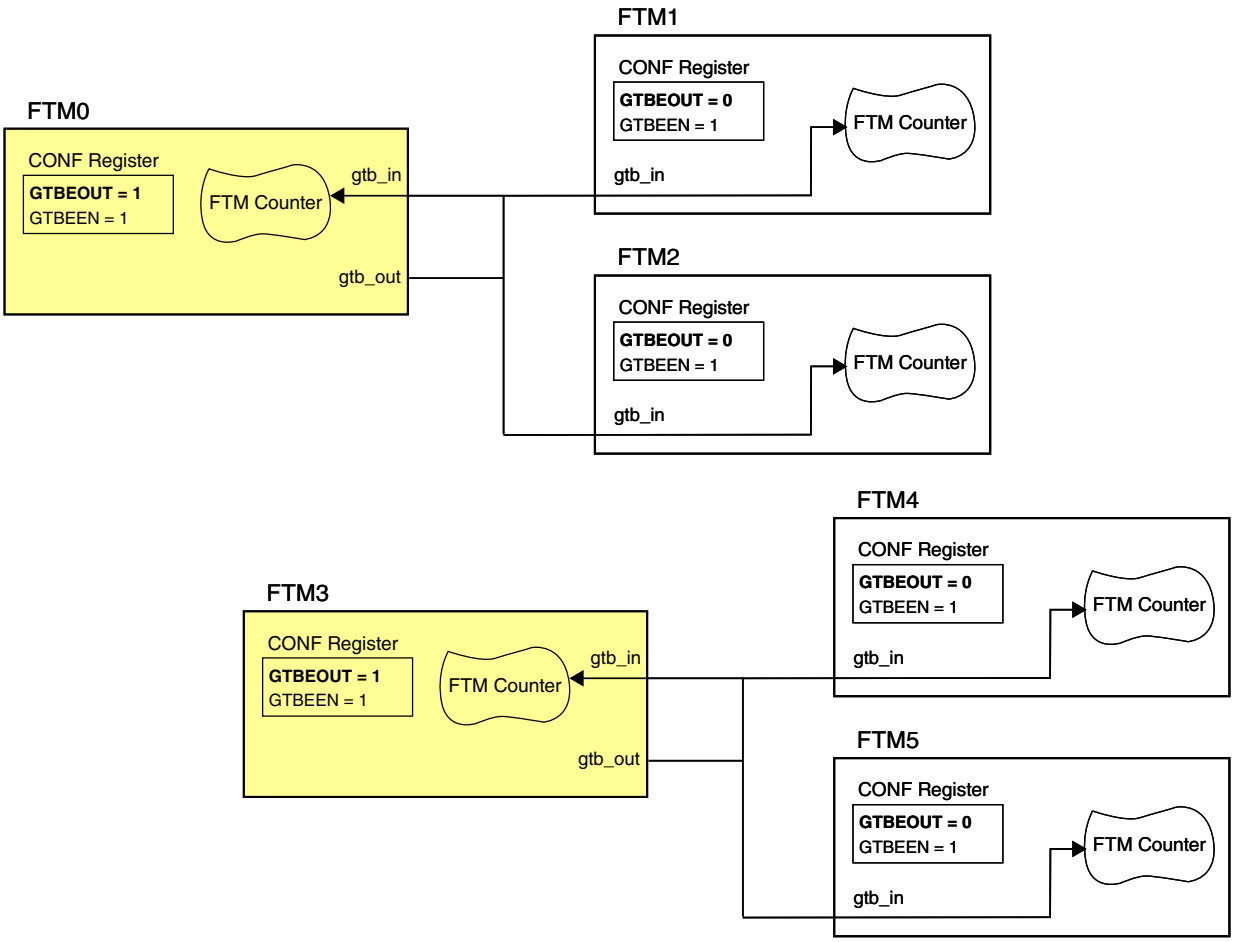


Figure 3-29. FTM Global Time Base Configuration

3.9.1.12 FTM BDM and debug halt mode

In the FTM chapter, references to the chip being in "BDM" are the same as the chip being in “debug halt mode".

3.9.2 Low-power timer configuration

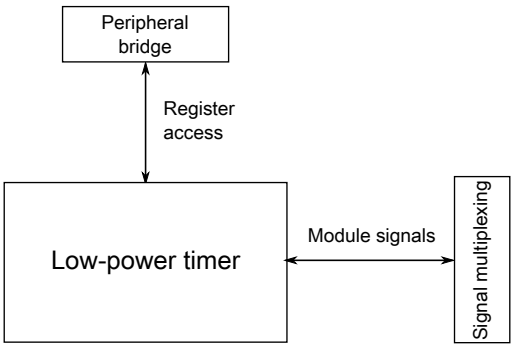


Figure 3-30. LPTMR configuration

Table 3-48. Reference links to related information

Topic	Related module	Reference
Full description	Low-power timer	Low-power timer
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal Multiplexing	Port control	Signal Multiplexing

3.9.2.1 LPTMR Instantiation Information

The low-power timer (LPTMR) allows operation during all power modes. The LPTMR can operate as a real-time interrupt or pulse accumulator. It includes a 15-bit prescaler (real-time interrupt mode) or glitch filter (pulse accumulator mode).

The LPTMR can be clocked from the internal reference clock, the internal 1 kHz LPO, OSCERCLK, or an external 32.768 kHz crystal.

An interrupt is generated (and the counter may reset) when the counter equals the value in the 16-bit compare register.

3.9.2.2 LPTMR pulse counter input options

The LPTMR_CSR[TPS] bitfield configures the input source used in pulse counter mode. The following table shows the chip-specific input assignments for this bitfield.

LPTMR_CSR[TPS]	Pulse counter input number	Chip input
00	0	CMP0 output
01	1	LPTMR0_ALT1 pin
10	2	LPTMR0_ALT2 pin
11	3	LPTMR0_ALT3 pin

3.9.2.3 LPTMR prescaler/glitch filter clocking options

The prescaler and glitch filter of the LPTMR module can be clocked from one of four sources determined by the LPTMR0_PSR[PCS] bitfield. The following table shows the chip-specific clock assignments for this bitfield.

NOTE

The chosen clock must remain enabled if the LPTMR is to continue operating in all required low-power modes.

LPTMR0_PSR[PCS]	Prescaler/glitch filter clock number	Chip clock
00	0	MCGIRCLK — internal reference clock (not available in VLLS modes)
01	1	LPO — 1 kHz clock (not available in VLLS0 mode)
10	2	ERCLK32K (not available in VLLS0 mode when using 32 kHz oscillator)
11	3	OSCERCLK — external reference clock

See Clock Distribution for more details on these clocks.

3.9.3 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.

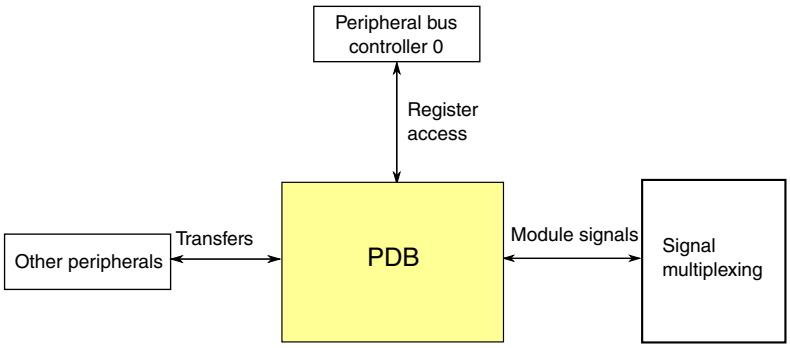


Figure 3-31. PDB configuration

Table 3-49. 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	Port control	Signal multiplexing

3.9.3.1 PDB Instantiation

3.9.3.1.1 PDB Output Triggers

Table 3-50. PDB0 output triggers

Number of PDB channels for ADC trigger	2
Number of pre-triggers per PDB channel	2
Number of DAC triggers	1
Number of PulseOut	2

Table 3-51. PDB1 output triggers

Number of PDB channels for ADC trigger	2
Number of pre-triggers per PDB channel	2
Number of DAC triggers	1
Number of PulseOut	2

3.9.3.1.2 PDB Input Trigger Connections

Table 3-52. PDB0 Input Trigger Options

PDB Trigger	PDB Input
0000	PDB_EXTRG0
0001	CMP 0
0010	CMP 1
0011	PDB_EXTRG1
0100	DMA Ch 0 Transfer Done
0101	DMA Ch 1 Transfer Done
0110	DMA Ch 2 Transfer Done
0111	DMA Ch 3 Transfer Done
1000	FTM0 Init and Ext Trigger Outputs
1001	FTM1 Init and Ext Trigger Outputs
1010	FTM2 Init and Ext Trigger Outputs
1011	FTM3 Init and Ext Trigger Outputs
1100	FTM4 Init and Ext Trigger Outputs
1101	FTM5 Init and Ext Trigger Outputs
1110	LPTMR Output
1111	Software Trigger

Table 3-53. PDB1 Input Trigger Options

PDB Trigger	PDB Input
0000	PDB_EXTRG0
0001	CMP 0
0010	CMP 1
0011	PDB_EXTRG1
0100	DMA Ch 4 Transfer Done
0101	DMA Ch 5 Transfer Done
0110	DMA Ch 6 Transfer Done
0111	DMA Ch 7 Transfer Done
1000	FTM0 Init and Ext Trigger Outputs
1001	FTM1 Init and Ext Trigger Outputs
1010	FTM2 Init and Ext Trigger Outputs
1011	FTM3 Init and Ext Trigger Outputs
1100	FTM4 Init and Ext Trigger Outputs
1101	FTM5 Init and Ext Trigger Outputs
1110	LPTMR Output
1111	Software Trigger

3.9.3.2 PDB Module Interconnections

Table 3-54. PDB0 module interconnections

PDB trigger outputs	Connection
Channel 0 triggers	ADC0 trigger and DMA_MUX source 48
Channel 1 triggers	ADC1 trigger and TRIG1 of FTM0, FTM1, and FTM2
DAC triggers	DAC0
Pulse-out	Pulse-out connected to each CMP module's sample/window input to control sample operation

Table 3-55. PDB1 module interconnections

PDB trigger outputs	Connection
Channel 0 triggers	ADC0 trigger and DMA_MUX source 47
Channel 1 triggers	ADC1 trigger and TRIG1 of FTM3, FTM4, and FTM5
DAC triggers	DAC0
Pulse-out	Pulse-out connected to each CMP module's sample/window input to control sample operation

3.9.3.3 Back-to-back acknowledgement connections

The application code can set the PDB*x*_CH*n*C1[BB] bits to configure the PDB pre-triggers as a single chain or several chains.

3.9.3.4 PDB Interval Trigger Connections to DAC

In this MCU, PDB interval trigger connections to DAC are implemented as follows.

- Two PDB interval trigger 0 signals are ORed together and connected to DAC0 hardware trigger input.

3.9.3.5 DAC External Trigger Input Connections

In this MCU, the following DAC external trigger inputs are implemented.

- DAC external trigger input 0: ADC0_SC1A[COCO]

3.9.3.6 Pulse-Out Connection

Two PDB Pulse-Out signals are ORed together and connected to each CMP block and used for sample window.

3.9.3.7 Pulse-Out Enable Register Implementation

The following table shows the comparison of pulse-out enable register at the module and chip level.

Table 3-56. PDB pulse-out enable register

Register	Module implementation	Chip implementation
POnEN	7:0 - POEN 31:8 - Reserved	0 - POEN[0] for CMP0 1 - POEN[1] for CMP1 31:2 - Reserved

3.10 Communication interfaces

3.10.1 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.

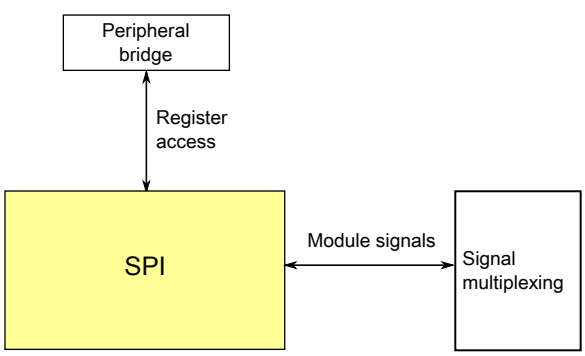


Figure 3-32. SPI configuration

Table 3-57. Reference links to related information

Topic	Related module	Reference
Full description	SPI	SPI
System memory map		System memory map

Table continues on the next page...

Table 3-57. Reference links to related information (continued)

Topic	Related module	Reference
Clocking		Clock Distribution
Signal Multiplexing	Port control	Signal Multiplexing

3.10.1.1 SPI Instantiation Information

This device contains one SPI module. It has 4-byte RX and TX FIFO, as well as DMA support.

3.10.1.2 SPI clocking

The SPI module is clocked by the system clock (the SPI refers to it as system clock). The module has an internal divider, with a minimum divide is 2.

So, the SPI can run at a maximum frequency of system clock/2 on PTD4, PTD5, PTD6, and PTD7 fast pads.

3.10.1.3 Number of CTARs

SPI CTAR registers define different transfer attribute configurations. The SPI module supports up to eight CTAR registers. This device supports two CTARs on the instances of the SPI.

In master mode, the CTAR registers define combinations of transfer attributes, such as frame size, clock phase, clock polarity, data bit ordering, baud rate, and various delays. In slave mode only CTAR0 is used, and a subset of its bitfields sets the slave transfer attributes.

3.10.1.4 TX FIFO size

Table 3-58. SPI transmit FIFO size

SPI Module	Transmit FIFO size
SPI0	4

3.10.1.5 RX FIFO Size

SPI supports up to 16-bit frame size during reception.

Table 3-59. SPI receive FIFO size

SPI Module	Receive FIFO size
SPI0	4

3.10.1.6 Number of PCS signals

The following table shows the number of peripheral chip select signals available per SPI module.

Table 3-60. SPI PCS signals

SPI Module	PCS Signals
SPI0	SPI_PCS[4:0]

3.10.1.7 SPI Operation in Low Power Modes

In VLPR and VLPW modes the SPI is functional; however, the reduced system frequency also reduces the max frequency of operation for the SPI. In VLPR and VLPW modes the max SPI_CLK frequency is 1 MHz.

In stop and VLPS modes, the clocks to the SPI module are disabled. The module is not functional, but it is powered so that it retains state.

There is one way to wake from stop mode via the SPI, which is explained in the following section.

3.10.1.8 Using GPIO Interrupt to Wake from stop mode

Here are the steps to use a GPIO to create a wakeup upon reception of SPI data in slave mode:

1. Point the GPIO interrupt vector to the desired interrupt handler.
2. Enable the GPIO input to generate an interrupt on either the rising or falling edge (depending on the polarity of the chip select signal).
3. Enter Stop or VLPS mode and Wait for the GPIO interrupt.

NOTE

It is likely that in using this approach the first word of data from the SPI host might not be received correctly. This is dependent on the transfer rate used for the SPI, the delay between chip select assertion and presentation of data, and the system interrupt latency.

3.10.1.9 SPI Doze Mode

The Doze mode for the SPI module is the same as the Wait and VLPW modes for the chip.

3.10.1.10 SPI Interrupts

The SPI has multiple sources of interrupt requests. However, these sources are OR'd together to generate a single interrupt request to the interrupt controller. When an SPI interrupt occurs, read the SPI_SR to determine the exact interrupt source.

3.10.1.11 SPI clocks

This table shows the SPI module clocks and the corresponding chip clocks.

Table 3-61. SPI clock connections

Module clock	Chip clock
System Clock	System Clock

3.10.2 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.

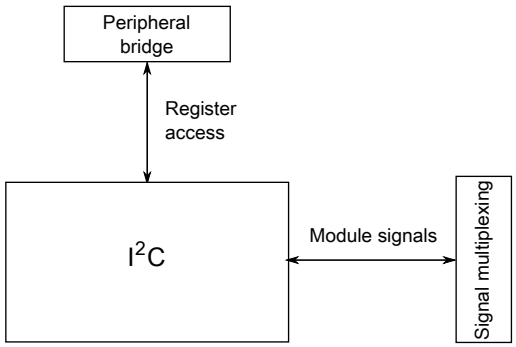


Figure 3-33. I2C configuration

Table 3-62. Reference links to related information

Topic	Related module	Reference
Full description	I2C	I2C
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal Multiplexing	Port control	Signal Multiplexing

3.10.2.1 IIC Instantiation Information

This device has one IIC module.

When the package pins associated with IIC have their mux select configured for IIC operation, the pins (SCL and SDA) are driven in a pseudo open drain configuration.

NOTE

Pseudo open drain means that the pins are not true open drain. The IIC SDA and SCL pins will drive a logic 0 (connect to VSS) and release a logic 1 as tri-state. A pull-up resistor tied to VDD is connected to SDA and SCL lines to ensure a reliable logic 1. The SDA and SCL pins cannot be connected to a pull-up resistor that is higher than the VDD of the device.

3.10.3 UART 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.

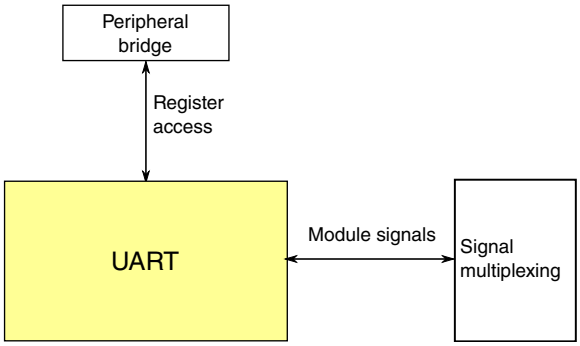


Figure 3-34. UART configuration

Table 3-63. Reference links to related information

Topic	Related module	Reference
Full description	UART	UART
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Signal Multiplexing	Port control	Signal Multiplexing

3.10.3.1 UART configuration

This device contains two UART modules. This section describes how each module is configured on this device.

1. Standard features of all UARTs:
   - RS-485 support
   - Hardware flow control (RTS/CTS)
   - 9-bit UART to support address mark with parity
   - MSB/LSB configuration on data
2. UART0 is clocked from the system clock. UART1 is clocked from the bus clock.
3. UART0 contains 8-entry transmit and 8-entry receive FIFOs.
4. UART1 contains a 1-entry transmit and receive FIFOs.

3.10.3.2 UART wakeup

The UART can be configured to generate an interrupt/wakeup on the first active edge that it receives.

3.10.3.3 UART interrupts

The UART has multiple sources of interrupt requests. However, some of these sources are OR'd together to generate a single interrupt request. See below for the mapping of the individual interrupt sources to the interrupt request:

The status interrupt combines the following interrupt sources:

Source	UART 0	UART 1
Transmit data empty	x	x
Transmit complete	x	x
Idle line	x	x
Receive data full	x	x
LIN break detect	x	x
RxD pin active edge	x	x
Initial character detect	x	

The error interrupt combines the following interrupt sources:

Source	UART 0	UART 1
Receiver overrun	x	x
Noise flag	x	x
Framing error	x	x
Parity error	x	x
Transmitter buffer overflow	x	x
Receiver buffer overflow	x	x
Receiver buffer underflow	x	x

3.10.4 FlexCAN 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.

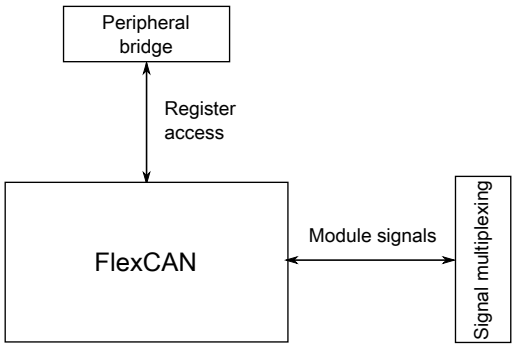


Figure 3-35. FlexCAN configuration

Table 3-64. Reference links to related information

Topic	Related module	Reference
Full description	FlexCAN0	FlexCAN
System memory map		System memory map
Clocking		Clocking

NOTE

The FlexCAN module is present only in KV11 parts.

3.10.4.1 Message buffer availability and effect on registers

This chip supports a maximum of 16 message buffers. As a result:

- In the IMASK1 and IFLAG1 registers, the only valid bit positions are 0 to 15.
- The maximum supported value of CTRL2[RFFN] is 4h.

3.10.4.2 FlexCAN glitch filter

This chip supports wakeup from the FlexCAN module's Stop and Wait mode through a CAN wake-up interrupt. Any recessive to dominant transition on the CAN bus (CAN_RX) can wake the chip from Stop or Wait 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 wake-up source, either from the filtered or unfiltered CAN bus
- Routing of the wake-up source to either the synchronous (Wait) 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 4 MHz ROSC. The reference clock for the glitch filter is targeted to be 4 MHz clock derived from the on-chip 4 MHz ROSC within the MCG. The user must ensure that the fast internal oscillator is selected MCG_C2[IRCS] and the MCG_SC[FCRDIV] bits. The glitch filter counts 11 cycles of the 4 MHz clock before recognizing it as a valid recessive to dominant transition.

3.11 Human-machine interfaces (HMI)

3.11.1 GPIO Configuration

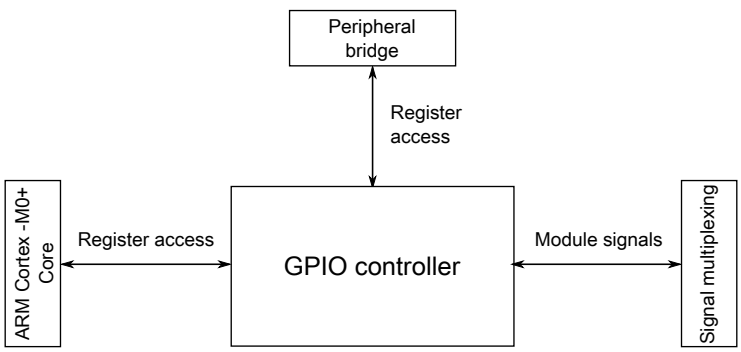


Figure 3-36. GPIO configuration

Table 3-65. Reference links to related information

Topic	Related module	Reference
Full description	GPIO	GPIO
System memory map		System memory map
Clocking		Clock Distribution
Power management		Power management
Crossbar switch	Crossbar switch	Crossbar switch
Signal Multiplexing	Port control	Signal Multiplexing

3.11.1.1 GPIO Instantiation Information

The device includes eight pins, PTB0, PTB1, PTC3, PTC4, PTD4, PTD5, PTD6, and PTD7, with high current drive capability out of which PTD4, PTD5, PTD6, and PTD7 also support high-speed capability. These pins can be used to drive LED or power MOSFET directly. The high drive capability applies to all functions which are multiplexed on these pins (UART, FTM, SPI...etc)

PTC6 and PTC7 are true open drain pins. These pins cannot drive a logic one without a pull-up resistor.

3.11.1.1.1 Pull Devices and Directions

The pull devices are enabled out of POR only on RESET_B, NMI_b and respective SWD signals. Other pins can be enabled by writing to PORTx_PCRn[PE] field.

All the pins have controllable pull direction using PORTx_PCRn[PS]. When enabled, all the pins default to pullup except for SWD_CLK.

3.11.1.2 GPIO accessibility in the memory map

The GPIO is multi-ported and can be accessed directly by the core with zero wait states at base address 0xF800_0000. It can also be accessed by the core and DMA masters through the cross bar/AIPS interface at 0x400F_F000 and at an aliased slot (15) at address 0x4000_F000. All BME operations to the GPIO space can be accomplished referencing the aliased slot (15) at address 0x4000_F000. Only some of the BME operations can be accomplished referencing GPIO at address 0x400F_F000.

3.12 Kinetis Motor Suite Configuration

3.12.1 KMS configuration

KMS is an integrated solution of hardware, factory programmed embedded firmware, and PC software that enables rapid development of applications driven by three phase permanent magnet or brushless DC motors.

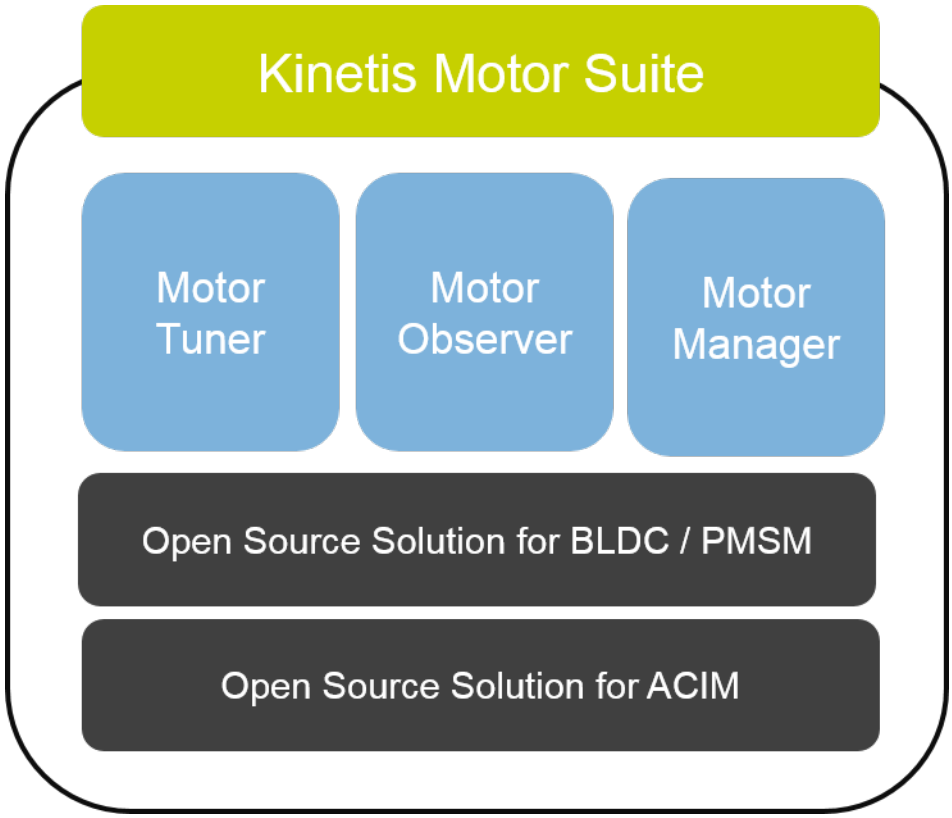


Figure 3-37. Kinetis Motor Suite

MCU's enabled with KMS have only 120/56K of flash available since 8K of flash is used to store the KMS library routines. For more information refer to Kinetis Motor Suite API Reference Manual (KMSRM)[3] and Kinetis Motor Suite User's Guide (KMSUG)[3].

3.12.2 Configuration of the production MCUs

The production KMS enabled MCUs comes with the following configuration:

- KMS library preprogrammed into the top 8 KB of flash
- The Flash Access Controls (FAC) XACCL0, XACCL1, XACCL2 and XACCL3 pre-programmed so they are not available for any future use.
- The Kinetis Flashloader is programmed into the MCU flash for one-time use. See Chapter 13 Kinetis Flashloader for more details
- The Flash protection bits are set to protect only the top 8 KB of flash. FPROT0, FPROT1, FPROT2, and FPROT3.
- The FSEC register configuration is set to disable mass erase of the flash and to leave the MCU unsecure. FSEC = 0xEE

3. To find the associated resource, go to http://www.nxp.com and perform a search using Document ID

With these setting a production programming tool cannot execute a "mass erase" command from the flashloader or via the SWD/JTAG debug interface. However, the "erase region" command or "erase flash sector" command can be used to sequentially erase all but the last 8 KB of flash. Using sector erase is a bit slower but it protects the KMS library from erasure since the FAC protection bits will not allow the execute only region to be sector erased.

If the Flash configuration space is erased and re-programmed with different settings of the FPROTx and FSEC registers that allow mass erase then subsequent attempts to perform a mass erase would be successful and the KMS library would be erased. Unfortunately, the only way to replace the erased KMS library is to replace the MCU chip with another production chip with the pre-programmed KMS library.

For more information refer to Kinetis Motor Suite API Reference Manual (KMSRM), and Kinetis Motor Suite User's Guide (KMSUG).

3.12.3 KMS Library

NOTE

Do not mass erase the MCU Flash. Doing a mass erase will remove the factory programmed secured library from the MCU and render the KMS enabled part obsolete for KMS use. If the part is accidentally mass erased and KMS is desired, then a new KMS enabled MCU is needed to replace the erased MCU. To prevent mass erase set the MEEN bits in the FSEC config to (FSEC[MEEN] = 10). Regardless of the SEC bits values, setting (FSEC[MEEN] = 10) will keep the MCU from being mass erased from the MCU 'erase all' command, the debugger write to MDM-AP register.

3.12.4 Library Protection

The KMS library is protected from reading through the flash access controller of the Flash module. The reserved address region is from 0x0001E000 to 0x0001FFFF on 128K Flash part and 0x0000E000 to 0x0000FFFF on 64K Flash part.

The Flash access controls are fully utilized by KMS for this protection and are not available to the user to protect any other code. If code protection is desired then use the other flash protection options available.

3.12.5 Flash protection

The KMS library can be protected from sector erase with the FPROT config bits. Setting the highest order bits of the FPROT register protects the flash commands Erase all blocks and Erase All Execute-only Segments.

See Using Kinetis Security and Protection Features (AN4507) for more information.

Chapter 4
Memory Map

4.1 Introduction

This device contains various memories and memory-mapped peripherals which are located in a 4 GB memory space. This chapter describes the memory and peripheral locations within that memory space.

4.2 System memory map

The following table shows the high-level device memory map.

Table 4-1. System memory map

System 32-bit Address Range	Destination Slave	Access
0x0000_0x0000–0x01_FFFF[1]	128 KB program flash and read-only data (Includes exception vectors in first 196 bytes)	All masters Cortex-M0+ core
0x0002_0000–0x1FFF_F7FFF	Reserved	—
0x1FFF_F000-0x1FFF_FFFF[2]	SRAM_L: Lower SRAM (4K)	All masters Cortex M0+ core
0x2000_0000-0x2000_2FFF[2]	SRAM_U: Upper SRAM (12K)	All masters Cortex M0+ core
0x2000_3000–0x3FFF_FFFF	Reserved	–
0x4000_0000–0x4007_FFFF	AIPS Peripherals	Cortex-M0+ core & DMA
0x4008_0000–0x400F_EFFF	Reserved	–
0x400F_F000–0x400F_FFFF	General purpose input/output (GPIO)	Cortex-M0+ core & DMA
0x4010_0000–0x43FF_FFFF	Reserved	–
0x4400_0000–0x5FFF_FFFF	Bit Manipulation Engine (BME) access to AIPS Peripherals for slots 0-127[3]	Cortex-M0+ core
0x6000_0000–0xDFFF_FFFF	Reserved	–
0xE000_0000–0xE00F_FFFF	Private Peripherals	Cortex-M0+ core

Table continues on the next page...

Table 4-1. System memory map (continued)

System 32-bit Address Range	Destination Slave	Access
0xE010_0000–0xEFFF_FFFF	Reserved	–
0xF000_0000–0xF000_0FFF	Micro Trace Buffer (MTB) registers	Cortex-M0+ core
0xF000_1000–0xF000_1FFF	MTB Data Watchpoint and Trace (MTBDWT) registers	Cortex-M0+ core
0xF000_2000–0xF000_2FFF	ROM table	Cortex-M0+ core
0xF000_3000–0xF000_3FFF	Miscellaneous Control Module (MCM)	Cortex-M0+ core
0xF000_4000–0xF000_4FFF	Memory Mapped Divide and Square Root (MMDVSQ)	Cortex-M0+ core
0xF000_5000–0xF7FF_FFFF	Reserved	–
0xF800_0000–0xFFFF_FFFF	IOPORT: GPIO (single cycle)	Cortex-M0+ core

1. The program flash always begins at 0x0000_0000 but the end of implemented flash varies depending on the amount of flash implemented for a particular device. See Flash Memory Sizes for details.
2. This range varies depending on SRAM sizes. See SRAM Ranges for details.
3. Includes BME operations to GPIO at slot 15 (based at 0x4000_F000).

4.3 Flash Memory Map

The flash memory and the flash registers are located at different base addresses as shown in the following figure. The base address for each is specified in System memory map.

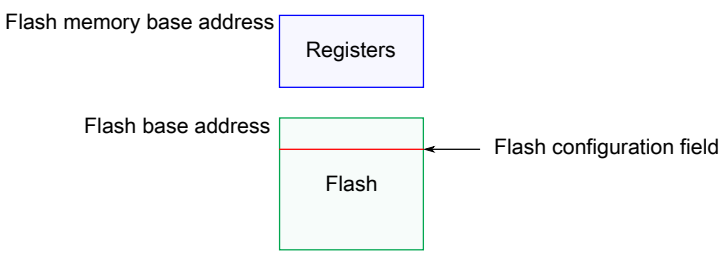


Figure 4-1. Flash memory map

The on-chip Flash is implemented in a portion of the allocated Flash range to form a contiguous block in the memory map beginning at address 0x0000_0000. See Flash memory sizes for details of supported ranges.

Accesses to the flash memory ranges outside the amount of Flash on the device causes the bus cycle to be terminated with an error followed by the appropriate response in the requesting bus master. Read collision events in which flash memory is accessed while a flash memory resource is being manipulated by a flash command also generates a bus error response.

4.3.1 Alternate Non-Volatile IRC User Trim Description

The following non-volatile locations (4 bytes) are reserved for custom IRC user trim supported by some development tools. An alternate IRC trim to the factory loaded trim can be stored at this location. To override the factory trim, user software must load new values into the MCG trim registers.

Non-Volatile Byte Address	Alternate IRC Trim Value
0x0000_03FC	Reserved
0x0000_03FD	Reserved
0x0000_03FE (bit 0)	SCFTRIM
0x0000_03FE (bit 4:1)	FCTRIM
0x0000_03FE (bit 5)	FCFTRIM
0x0000_03FF	SCTRIM

4.4 SRAM memory map

The on-chip RAM is split between SRAM_L and SRAM_U. The RAM is also implemented such that the SRAM_L and SRAM_U ranges form a contiguous block in the memory map. See SRAM ranges for details.

Accesses to the SRAM_L and SRAM_U memory ranges outside the amount of RAM on the device causes the bus cycle to be terminated with an error followed by the appropriate response in the requesting bus master.

4.5 Bit Manipulation Engine

The Bit Manipulation Engine (BME) provides hardware support for atomic read-modify-write memory operations to the peripheral address space. By combining the basic load and store instruction support in the Cortex-M instruction set architecture with the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. See the Bit Manipulation Engine Block Guide (BME) for a detailed description of BME functionality.

4.6 Peripheral bridge (AIPS-Lite) memory map

The peripheral memory map is accessible via one slave port on the crossbar in the 0x4000_0000–0x400F_FFFF region. The device implements one peripheral bridge that defines a 1024 KB address space.

The three regions associated with this space are:

- A 128 KB region, partitioned as 32 spaces, each 4 KB in size and reserved for on-platform peripheral devices. The AIPS controller generates unique module enables for all 32 spaces.
- A 384 KB region, partitioned as 96 spaces, each 4 KB in size and reserved for off-platform modules. The AIPS controller generates unique module enables for all 96 spaces.
- The last slot is a 4 KB region beginning at 0x400F_F000 for accessing the GPIO module. The GPIO slot (slot 128) is an alias of slot 15. This block is also directly interfaced to the core and provides direct access without incurring wait states associated with accesses via the AIPS controller.

Modules that are disabled via their clock gate control bits in the SIM registers disable the associated AIPS slots. Access to any address within an unimplemented or disabled peripheral bridge slot results in a transfer error termination.

For programming model accesses via the peripheral bridges, there is generally only a small range within the 4 KB slots that is implemented. Accessing an address that is not implemented in the peripheral results in a transfer error termination.

4.6.1 Read-after-write sequence and required serialization of memory operations

In some situations, a write to a peripheral must be completed fully before a subsequent action can occur. Examples of such situations include:

- Exiting an interrupt service routine (ISR)
- Changing a mode
- Configuring a function

In these situations, the application software must perform a read-after-write sequence to guarantee the required serialization of the memory operations:

1. Write the peripheral register.
2. Read the written peripheral register to verify the write.
3. Continue with subsequent operations.

4.6.2 Peripheral Bridge (AIPS-Lite) Memory Map

Table 4-2. Peripheral bridge 0 slot assignments

System 32-bit base address	Slot number	Module
0x4000_0000	0	—
0x4000_1000	1	—
0x4000_2000	2	—
0x4000_3000	3	—
0x4000_4000	4	—
0x4000_5000	5	—
0x4000_6000	6	—
0x4000_7000	7	—
0x4000_8000	8	DMA controller
0x4000_9000	9	DMA controller transfer control descriptors
0x4000_A000	10	—
0x4000_B000	11	—
0x4000_C000	12	—
0x4000_D000	13	—
0x4000_E000	14	—
0x4000_F000	15	GPIO controller (aliased to 0x400F_F000)
0x4001_0000	16	—
0x4001_1000	17	—
0x4001_2000	18	—
0x4001_3000	19	—
0x4001_4000	20	—
0x4001_5000	21	—
0x4001_6000	22	—
0x4001_7000	23	—
0x4001_8000	24	—
0x4001_9000	25	—
0x4001_A000	26	—
0x4001_B000	27	—
0x4001_C000	28	—
0x4001_D000	29	—
0x4001_E000	30	—
0x4001_F000	31	—
0x4002_0000	32	Flash memory
0x4002_1000	33	DMA channel mutiplexer 0
0x4002_2000	34	—

Table continues on the next page...

Table 4-2. Peripheral bridge 0 slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4002_3000	35	—
0x4002_4000	36	FlexCAN0
0x4002_5000	37	—
0x4002_6000	38	FlexTimer 3 (FTM3) - 6 channel
0x4002_7000	39	FlexTimer 4 (FTM4) - 2 channel
0x4002_8000	40	FlexTimer 5 (FTM5) - 2 channel
0x4002_9000	41	—
0x4002_A000	42	—
0x4002_B000	43	—
0x4002_C000	44	SPI0
0x4002_D000	45	—
0x4002_E000	46	—
0x4002_F000	47	—
0x4003_0000	48	—
0x4003_1000	49	PDB1
0x4003_2000	50	CRC
0x4003_3000	51	—
0x4003_4000	52	—
0x4003_5000	53	—
0x4003_6000	54	PDB0
0x4003_7000	55	—
0x4003_8000	56	FlexTimer 0 (FTM0) - 6 channel
0x4003_9000	57	FlexTimer 1 (FTM1) - 2 channel
0x4003_A000	58	FlexTimer 2 (FTM2) - 2 channel
0x4003_B000	59	Analog-to-digital converter (ADC0)
0x4003_C000	60	Analog-to-digital converter (ADC1)
0x4003_D000	61	—
0x4003_E000	62	—
0x4003_F000	63	DAC0
0x4004_0000	64	Low-power timer (LPTMR)
0x4004_1000	65	—
0x4004_2000	66	—
0x4004_3000	67	—
0x4004_4000	68	—
0x4004_5000	69	—
0x4004_6000	70	—
0x4004_7000	71	SIM low-power logic
0x4004_8000	72	System integration module (SIM)
0x4004_9000	73	Port A multiplexing control

Table continues on the next page...

Table 4-2. Peripheral bridge 0 slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4004_A000	74	Port B multiplexing control
0x4004_B000	75	Port C multiplexing control
0x4004_C000	76	Port D multiplexing control
0x4004_D000	77	Port E multiplexing control
0x4004_E000	78	—
0x4004_F000	79	—
0x4005_0000	80	—
0x4005_1000	81	—
0x4005_2000	82	Watchdog timer (gen2008)
0x4005_3000	83	—
0x4005_4000	84	—
0x4005_5000	85	—
0x4005_6000	86	—
0x4005_7000	87	—
0x4005_8000	88	—
0x4005_9000	89	—
0x4005_A000	90	—
0x4005_B000	91	—
0x4005_C000	92	—
0x4005_D000	93	—
0x4005_E000	94	—
0x4005_F000	95	—
0x4006_0000	96	—
0x4006_1000	97	External Watchdog Monitor (EWM)
0x4006_2000	98	—
0x4006_3000	99	—
0x4006_4000	100	Multi-purpose Clock Generator (MCG)
0x4006_5000	101	System oscillator (OSC)
0x4006_6000	102	I^2C 0
0x4006_7000	103	—
0x4006_8000	104	—
0x4006_9000	105	—
0x4006_A000	106	UART 0
0x4006_B000	107	UART 1
0x4006_C000	108	—
0x4006_D000	109	—
0x4006_E000	110	—
0x4006_F000	111	—
0x4007_0000	112	—

Table continues on the next page...

Table 4-2. Peripheral bridge 0 slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4007_1000	113	—
0x4007_2000	114	—
0x4007_3000	115	Analog comparator (CMP) / 6-bit digital-to-analog converter (DAC)
0x4007_4000	116	—
0x4007_5000	117	—
0x4007_6000	118	—
0x4007_7000	119	—
0x4007_8000	120	—
0x4007_9000	121	—
0x4007_A000	122	—
0x4007_B000	123	—
0x4007_C000	124	Low-leakage wakeup unit (LLWU)
0x4007_D000	125	Power management controller (PMC)
0x4007_E000	126	System Mode controller (SMC)
0x4007_F000	127	Reset Control Module (RCM)
0x400F_F000	128	GPIO controller

4.6.3 Modules Restricted Access in User Mode

In user mode, for MCG, RCM, SIM (slot 71 and 72), SMC, LLWU, and PMC, reads are allowed, but writes are blocked and generate bus error.

4.7 Private Peripheral Bus (PPB) memory map

The PPB is part of the defined ARM bus architecture and provides access to select processor-local modules. These resources are only accessible from the core; other system masters do not have access to them.

Table 4-3. PPB memory map

System 32-bit Address Range	Resource	Additional Range Detail	Resource
0xE000_0000–0xE000_DFFF	Reserved		
0xE000_E000–0xE000_EFFF	System Control Space (SCS)	0xE000_E000–0xE000_E00F	Reserved
		0xE000_E010–0xE000_E0FF	SysTick
		0xE000_E100–0xE000_ECFF	NVIC
		0xE000_ED00–0xE000_ED8F	System Control Block

Table continues on the next page...

Table 4-3. PPB memory map (continued)

System 32-bit Address Range	Resource	Additional Range Detail	Resource
		0xE000_ED90–0xE000_EDEF	Reserved
		0xE000_EDF0–0xE000_EEFF	Debug
		0xE000_EF00–0xE000_EFFF	Reserved
0xE000_F000–0xE00F_EFFF	Reserved		
0xE00F_F000–0xE00F_FFFF	Core ROM Space (CRS)		
0xF000_4000–0xF000_4013	MMDVSQ		

Chapter 5
Clock Distribution

5.1 Introduction

This chapter includes the clock architecture for the device, the overview of the clocks, and a terminology section.

The Cortex M0+ resides within a synchronous core platform, where the processor and bus masters, flash, and peripheral clocks can be configured independently. The clock distribution figure shows how clocks from the MCG and XOSC modules are distributed to the microcontroller's other functional units. Some modules in the microcontroller have selectable clock input.

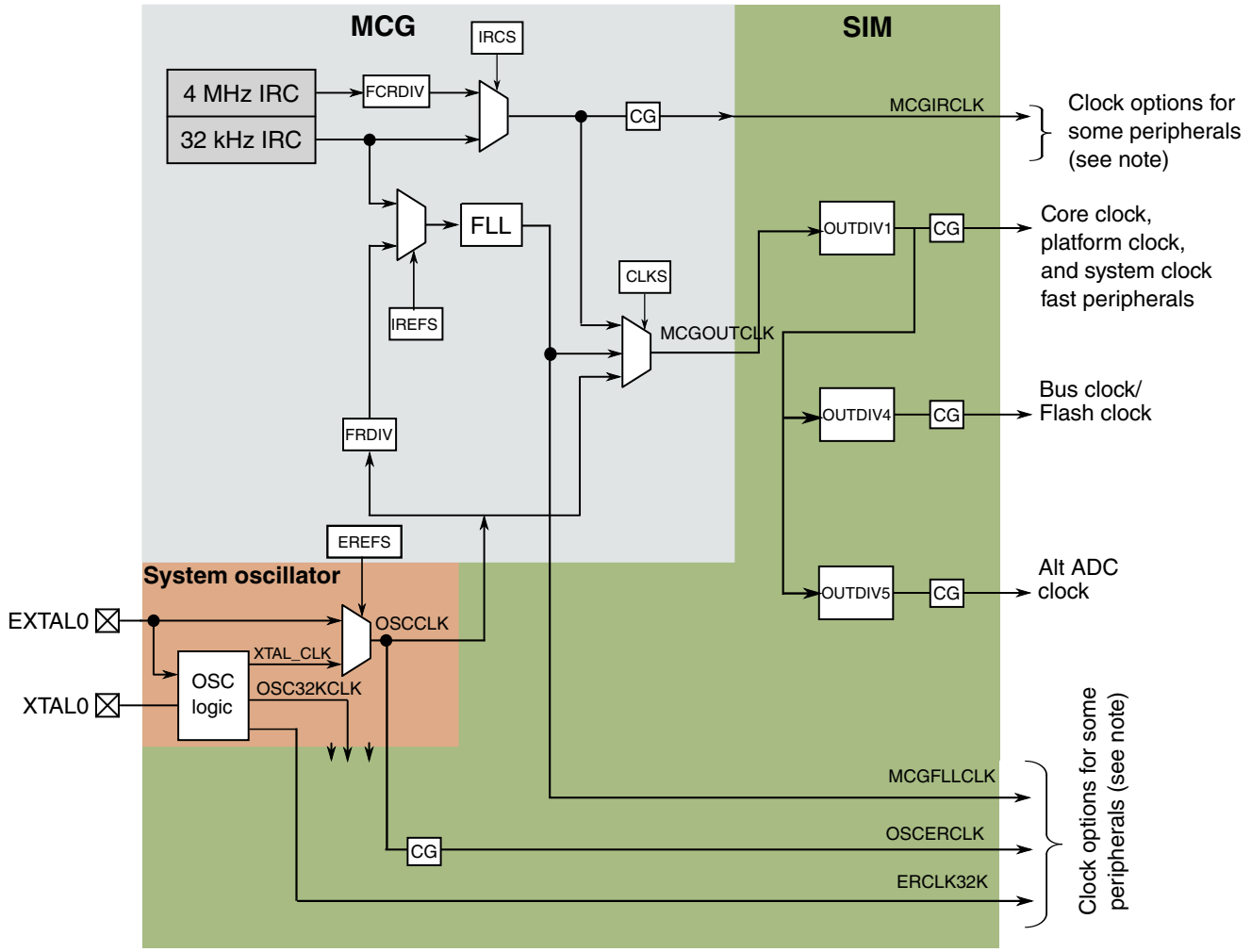
5.2 Programming model

The selection and multiplexing of system clock sources are controlled and programmed via the MCG module. The setting of clock dividers and module clock gating for the system is programmed via the SIM module. See those sections for detailed register and bit descriptions.

5.3 High-level device clocking diagram

The following system oscillator, MCG, and SIM module registers control the multiplexers, dividers, and clock gates shown in the following figure:

	OSC	MCG	SIM
Multiplexers	MCG_C*x*	MCG_C*x*	SIM_SOPT1, SIM_SOPT2
Dividers	—	MCG_C*x*	SIM_CLKDIV*x*
Clock gates	OSC_CR	MCG_C1	SIM_SCGC*x*



CG — Clock gate

Note: See subsequent sections for details on where these clocks are used.

Figure 5-1. Clocking diagram

5.4 Clock definitions

The following table describes the clocks in the previous block diagram.

Clock name	Description
Core clock	MCGOUTCLK divided by OUTDIV1, clocks the ARM Cortex-M0+ core
Platform clock	MCGOUTCLK divided by OUTDIV1, clocks the crossbar switch and NVIC
System clock	MCGOUTCLK divided by OUTDIV1, clocks the bus masters directly. Also drives high-speed peripherals FTM0, FTM1,FTM2, FTM3, FTM4, FTM5, FlexCAN0, SPI, and UART0.

Table continues on the next page...

Clock name	Description
Bus clock	System clock divided by OUTDIV4, clocks the bus slaves and peripherals.
Flash clock	Flash memory clock. On this device it is the same as the bus clock.
MCGIRCLK	MCG output of the slow or fast internal reference clock
MCGOUTCLK	MCG output of either IRC, MCGFLLCLK or MCG's external reference clock that sources the core, system, bus, and flash clock.
MCGFLLCLK	MCG output of the FLL. MCGFLLCLK may clock some modules. Additionally, this clock is used for UART0, and FTM modules.
OSCCLK	System oscillator output of the internal oscillator or sourced directly from EXTAL. Used as MCG external reference clock.
OSCERCLK	System oscillator output sourced from OSCCLK that may clock some on-chip modules
ALTADC_CLK	Alternative clock for ADC0 and ADC1 SAR conversion
LPO	PMC 1 kHz output
MCGFFCLK	MCG alternative output clock to FTM0, FTM1, FTM2, FTM3, FTM4, and FTM5

5.4.1 Device clock summary

The following table provides more information regarding the on-chip clocks.

Table 5-1. Clock Summary

Clock name	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when…
MCGOUTCLK	Up to 75 MHz	Up to 4 MHz	MCG	In all Stop modes except for Partial Stop modes
MCGFLLCLK	Up to 75 MHz	N/A	MCG	MCG clock controls do not enable and in all Stop modes
Core clock	Up to 75 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all Wait and Stop modes
Platform clock	Up to 75 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all Stop modes
System clock	Up to 75 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all Stop modes and Compute Operation
Bus clock	Up to 25 MHz	Up to 1 MHz [1]	MCGOUTCLK clock divider	In all Stop modes except for Partial Stop2 mode, and Compute Operation
SWD Clock	Up to 25 MHz	Up to 1 MHz	SWD_CLK pin	In all stop modes

Table continues on the next page...

Table 5-1. Clock Summary (continued)

Clock name	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when…
Flash clock	Up to 25 MHz	Up to 1 MHz in BLPE Up to 800 kHz in BLPI	MCGOUTCLK clock divider	In all stop modes except for partial STOP2 mode
Internal reference (MCGIRCLK)	30-40 kHz Slow IRC or 4 MHz Fast IRC	4 MHz Fast IRC only	MCG	MCG_C1[IRCLKEN] cleared, Stop/VLPS mode and MCG_C1[IREFSTEN] cleared, or VLLS mode
External reference (OSCERCLK)	Up to 48 MHz (bypass), 30-40 kHz or 3-32 MHz (crystal)	Up to 16 MHz (bypass), 30-40 kHz (low-range crystal) or 3-16 MHz (high-range crystal)	System OSC	System OSC's OSC_CR[ERCLKEN] cleared, or Stop mode and OSC_CR[EREFSTEN] cleared, or VLLS0 and oscillator not in external clock mode.
External reference 32kHz (ERCLK32K)	30-40 kHz	30-40 kHz	System OSC	System OSC's OSC_CR[ERCLKEN] cleared, or VLLS0 and oscillator not in external clock mode.
LPO	1 kHz	1 kHz	PMC	in VLLS0
ALTCLK	Up to 24 MHz	Up to 1 MHz or use internal RC clock.	MCGIRCLK, SIM_CLKDIV1[OUTDIV5], or OSCERCLK	There is no CGC bit to disable MCGIRCLK and OSCERCLK. SIM_CLKDIV1[OUTDIV5] can be disabled via SIM_CLKDIV1[OUTDIV5EN]

1. If in BLPI mode, where clocking is derived from the fast internal reference clock, the Bus clock and flash clock frequency needs to be limited to 800 kHz if executing from flash.

5.5 Internal clocking requirements

The clock dividers are programmed via the SIM module's CLKDIV registers. The following requirements must be met when configuring the clocks for this device:

1. The core, platform, and system clocks are programmable from a divide-by-1 through divide-by-16 setting. The core, platform, and system clock frequencies must be 75 MHz or slower.

2. The bus clock and flash clock frequencies are divided from the system clock and are programmable from a divide-by-1 through divide-by-8 setting. The bus clock and flash clock must be programmed to 25 MHz or slower.
3. The ALTADC_CLK must provide a frequency of less than 24 MHz to ensure the ADC provides a 1.1 μs or less conversion time. 24 MHz is the maximum clock speed for reliable ADC operation.
4. The system clock is also for FTM0, FTM1, FTM2, FTM3, FTM4, FTM5,FlexCAN0, SPI, and UART0. These peripherals can support high-speed operation and have internal prescalers to support slower clock operations.

The following is a common clock configuration for this device:

Clock	Frequency
Core clock	75 MHz
Platform clock	75 MHz
System clock	75 MHz
Bus and flash clock	25 MHz
ALTADC_CLK	24 MHz

5.5.1 Clock divider values after reset

Each clock divider is programmed via the SIM module's CLKDIV1 registers. Two bits in the flash memory's FTFA_FOPT register controls the reset value of the core clock, system clock, bus clock, flash clock, and alternative ADC clock with dividers as shown below:

FTFA_FOPT [4,0]	Core/system clock	Bus/Flash clock	Description
00	0x7 (divide by 8)	0x1 (divide by 2)	Low-power boot
01	0x3 (divide by 4)	0x1 (divide by 2)	Low-power boot
10	0x1 (divide by 2)	0x1 (divide by 2)	Low-power boot
11	0x0 (divide by 1)	0x1 (divide by 2)	Fast clock boot

This gives the user flexibility in selecting between a lower frequency, low-power boot option and a higher frequency, higher power option during and after reset.

The Flash Erased state defaults to Fast Clocking mode, because these bits reside in flash, which is logic 1 in the Flash Erased state. To enable a lower power boot option, program the appropriate bits in FTFA_FOPT. During the reset sequence, if either of the control bits is cleared, the system is in a slower clock configuration. On any system reset, the clock dividers return to this configurable reset state.

5.5.2 VLPR mode clocking

The clock dividers cannot be changed while in VLPR mode. They must be programmed prior to entering VLPR mode to guarantee operation. Maximum frequency limitations for VLPR mode are as follows :

- The core/system clocks are less than or equal to 4 MHz, and
- The bus and flash clocks are less than or equal to 1 MHz

NOTE

When the MCG is in BLPI and clocking is derived from the Fast IRC, the clock divider controls (MCG_SC[FCRDIV], SIM_CLKDIV1[OUTDIV1], and SIM_CLKDIV1[OUTDIV4]) must be programmed so that the resulting flash clock nominal frequency is 800 kHz or less. In this case, one example of correct configuration is MCG_SC[FCRDIV]=000b, SIM_CLKDIV1[OUTDIV1]=0000b, and SIM_CLKDIV1[OUTDIV4]=100b, resulting in a divide by 5 setting.

5.6 Clock Gating

The clock to each module can be individually gated on and off using the SIM module's SCGC*x* registers. These bits are cleared after any reset, which disables the clock to the corresponding module to conserve power. Prior to initializing a module, set the corresponding bit in the SCGC*x* register to enable the clock. Before disabling a clock to a specific module, disable the module to ensure that a predictable behavior occurs when the clock is enabled again.

Any bus access to a peripheral that has its clock disabled generates an error termination.

5.7 Module clocks

The following table summarizes the clocks associated with each module.

Table 5-2. Module clocks

Module	Bus interface clock	Internal clocks	I/O interface clocks
Core modules			

Table continues on the next page...

Table 5-2. Module clocks (continued)

Module	Bus interface clock	Internal clocks	I/O interface clocks
ARM Cortex-M0+ core	Platform clock	Core clock	—
NVIC	Platform clock	—	—
DAP	Platform clock	—	SWD_CLK
System modules			
DMA	System clock	—	—
DMA Mux	Bus clock	—	—
Port control	Bus clock	—	—
Crossbar Switch	Platform clock	—	—
Peripheral bridges	System clock	Bus clock	—
LLWU, PMC, SIM, RCM	Bus clock	LPO	—
Mode controller	Bus clock	—	—
MCM	Platform clock	—	—
EWM	Bus clock	LPO	—
Watchdog timer	Bus clock	LPO, MCGIRCLK	—
External Watchdog Monitor	Bus clock	LPO	
CRC	Bus clock	—	
Clocks			
MCG	Bus clock	MCGOUTCLK, MCGFLLCLK, MCGIRCLK, OSCERCLK	—
OSC	Bus clock	OSCERCLK	—
Memory and memory interfaces			
Flash Controller	Platform clock	Flash clock	—
Flash memory	Flash clock	—	—
Analog			
ADC0 / ADC1	Bus clock	ADC0ALTCLKSRC/ ADC1ALTCLKSRC, MCGIRCLK, OSCERCLK	—
CMP0 and CMP1	Bus clock	—	—
DAC	Bus clock	—	—
Timers			
FTM0, FTM1, FTM2, FTM3, FTM4, FTM5	System clock	MCGFFCLK, MCGIRCLK, OSCERCLK	FTM_CLKIN0, FTM_CLKIN1, FTM_CLKIN2
LPTMR	Bus clock	LPO, OSCERCLK, MCGIRCLK, ERCLK32K	—
PDB0 and PDB1	Bus clock	—	—
Communication interfaces			
SPI0	System clock	—	SPI0_SCK
I^2C0	Bus clock	—	I2C0_SCL
UART0	System clock	—	—
UART1	Bus clock	—	—
FlexCAN0	System clock	OSCERCLK, MCGIRCLK	—

Table continues on the next page...

Table 5-2. Module clocks (continued)

Module	Bus interface clock	Internal clocks	I/O interface clocks
Human-machine interfaces			
GPIO	Platform clock	—	—

5.7.1 PMC 1-kHz LPO clock

The Power Management Controller (PMC) generates a 1 kHz clock that is enabled in all modes of operation, including all low-power modes except VLLS0. This 1 kHz source is commonly referred to as the LPO clock or 1 kHz LPO clock.

5.7.2 WDOG clocking

The COP may be clocked from several clock sources as shown in the following figure. The Gen2008 watchdog can select one of 3 clock sources.

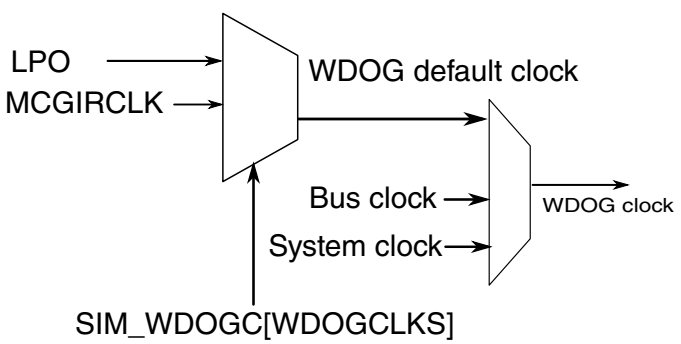


Figure 5-2. WDOG clock generation

5.7.3 LPTMR clocking

The prescaler and glitch filters in each of the LPTMR*x* modules can be clocked as shown in the following figure.

NOTE

The chosen clock must remain enabled if the LPTMR*x* is to continue operating in all required low-power modes.

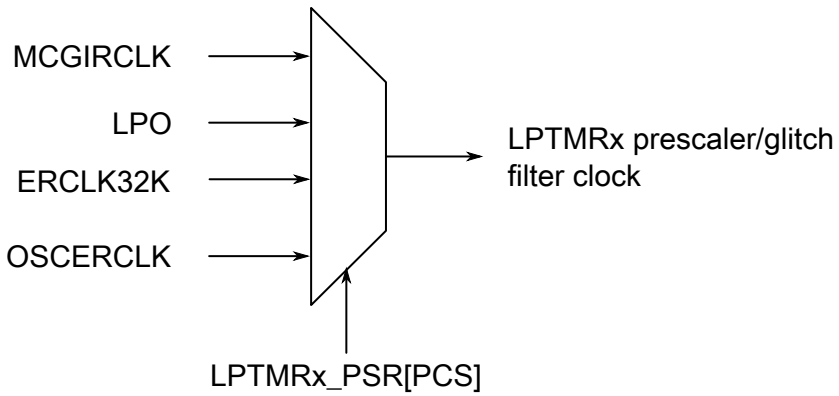


Figure 5-3. LPTMRx prescaler/glitch filter clock generation

5.7.4 FlexCAN clocking

The clock for the FlexCAN's protocol engine can be selected as shown in the following figure.

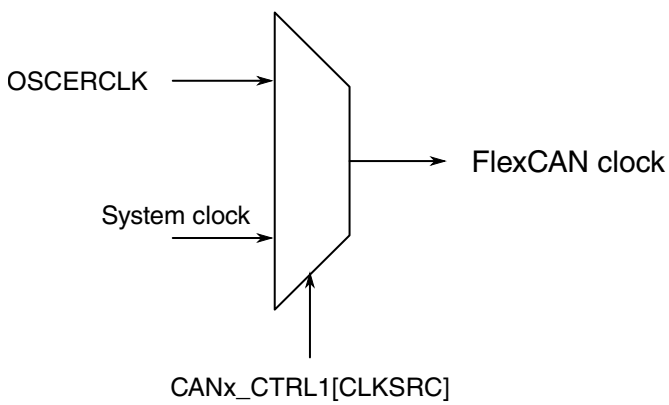


Figure 5-4. FlexCAN clocking diagram

5.7.5 Flex Timer (FTM) clocking

The Flextimers have the option to be clocked as shown below.

FTM0, FTM1, FTM2, FTM3, FTM4, and FTM5 are clocked by several clock sources. The FTMs each have three options: System clock, Fixed Frequency clock, and External clock. The system input clock to the FTMx can come from various sources, such as the MCG system clock, the MCGIRCLK, and the OSCERCLK. The external clock source for each FTM can come from three possible input pins, therefore providing the option to clock each FTM with a different clock frequency.

NOTE

The chosen clock must remain enabled if the FTM is to continue operating in all required low-power modes.

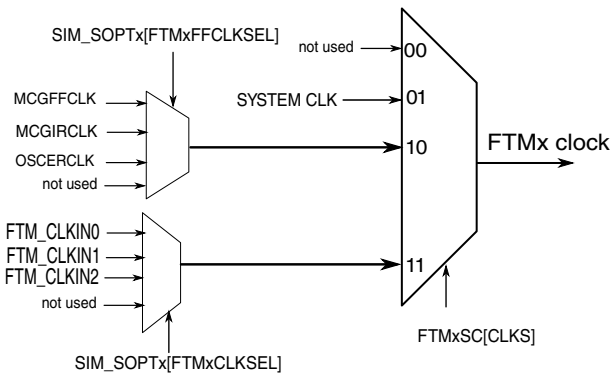


Figure 5-5. FlexTimer clock generation

5.7.6 UART clocking

The UART0 is clocked by the system clock and UART1 is clocked by the bus clock.

NOTE

The chosen clock must remain enabled if the UARTs are to continue operating in all required low-power modes.

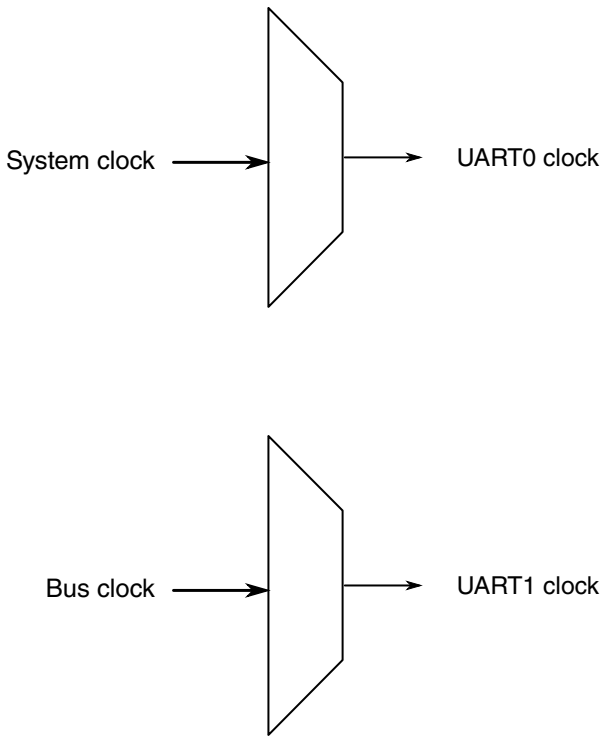


Figure 5-6. UARTx clock generation

5.7.7 SPI clocking

The SPI module has a selectable clock as shown in the following figure. The SPI is clocked by the system clock.

NOTE

The chosen clock must remain enabled if the SPI is to continue operating in all required low-power modes.

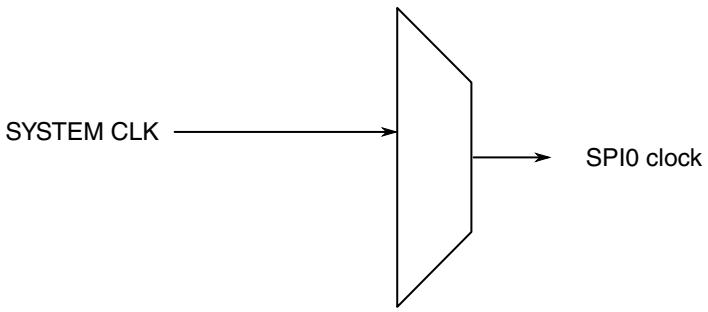


Figure 5-7. SPI0 clock generation

5.7.8 ADC clocking

The ADC0 and ADC1 modules have a selectable clock as shown in the following figure. The ADC is always clocked by the bus clock, but it has the additional option to clock the ADC SAR circuit with an alternative clock. This allows the ADC to have an independent divide from the CPU clock and ensure minimum conversion times can be fulfilled for various CPU frequencies.

NOTE

The chosen clock must remain enabled if the ADC is to continue operating in all required low-power modes.

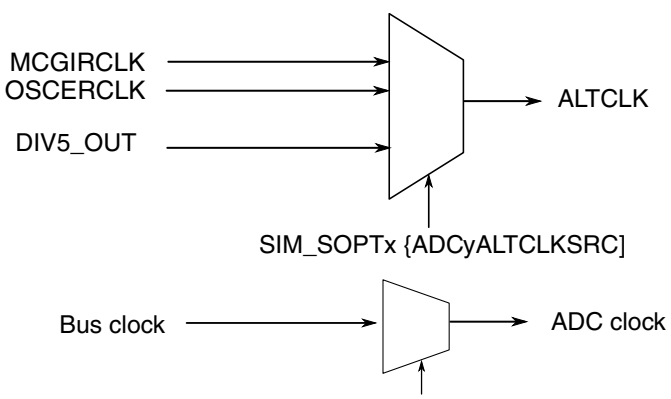


Figure 5-8. ADC0 and ADC1 clock generation

Chapter 6
Reset and Boot

6.1 Introduction

The following reset sources are supported in this MCU:

Table 6-1. Reset sources

Reset sources	Description
POR reset	• Power-on reset (POR)
System resets	• External pin reset (PIN) • Low-voltage detect (LVD) • Low leakage wakeup (LLWU) reset • Multipurpose clock generator loss of clock (LOC) reset • Stop mode acknowledge error (SACKERR) • Software reset (SW) • Lockup reset (LOCKUP) • MDM DAP system reset • Watchdog timer reset
Debug reset	• Debug reset

Each of the system reset sources has an associated bit in the system reset status (SRS) registers. See the Reset Control Module for register details.

The MCU can exit and reset in functional mode where the CPU is executing code (default) or the CPU is in a debug halted state. There are several boot options that can be configured. See Boot information for more details.

6.2 Reset

This section discusses basic reset mechanisms and sources. Some modules that cause resets can be configured to cause interrupts instead. Consult the individual peripheral chapters for more information.

6.2.1 Power-on reset (POR)

When power is initially applied to the MCU or when the supply voltage drops below the power-on reset re-arm voltage level (V_{POR}), the POR circuit causes a POR reset condition.

As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply has risen above the LVD low threshold (V_{LVDL}). The POR and LVD bits in reset status register are set following a POR.

6.2.2 System reset sources

Resetting the MCU provides a way to start processing from a known set of initial conditions. System reset begins with the on-chip regulator in full regulation and system clocking generation from an internal reference. When the processor exits reset, it performs the following:

- Reads the start SP (SP_main) from vector-table offset 0
- Reads the start PC from vector-table offset 4
- LR is set to 0xFFFF_FFFF

The on-chip peripheral modules are disabled and the non-analog I/O pins are initially configured as disabled. The pins with analog functions assigned to them default to their analog function after reset.

During and following a reset, the SWD pins have their associated input pins configured as:

- SWD_CLK in pull-down (PD)
- SWD_DIO in pull-up (PU)

6.2.2.1 External pin reset ($\overline{\text{RESET}}$)

This pin is open drain and has an internal pullup device. Asserting $\overline{\text{RESET}}$ wakes the device from any mode.

The $\overline{\text{RESET}}$ pin can be disabled by programming RESET_PIN_CFG option bit to 0. When this option selected, there could be a short period of contention during a POR ramp where the device drives the pin out low prior to establishing the setting of this option and releasing the RESET function on the pin.

6.2.2.1.1 Reset pin filter

The $\overline{\text{RESET}}$ pin filter supports filtering from both the 1 kHz LPO clock and the bus clock. The RPFC[RSTFLTSS], RPFC[RSTFLTSRW], and RPFW[RSTFLTSEL] fields in the reset control (RCM) register set control this functionality; see the RCM chapter. The filters are asynchronously reset by Chip POR. The reset value for each filter assumes the RESET pin is negated.

For all stop modes where LPO clock is still active (Stop, VLPS, VLLS3, and VLLS1), the only filtering option is the LPO based digital filter. The filtering logic either switches to bypass operation or has continued filtering operation depending on the filtering mode selected. When entering VLLS0, the RESET pin filter is disabled and bypassed.

The LPO filter has a fixed filter value of 3. Due to a synchronizer on the input data, there is also some associated latency (2 cycles). As a result, 5 cycles are required to complete a transition from low to high or high to low.

6.2.2.2 Low-voltage detect (LVD)

The chip includes a system for managing low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system consists of a power-on reset (POR) circuit and an LVD circuit with a user-selectable trip voltage. The LVD system is always enabled in normal run, wait, or stop mode. The LVD system is disabled when entering VLPx or VLLSx modes.

The LVD can be configured to generate a reset upon detection of a low voltage condition by setting the PMC's LVDSC1[LVDRE] bit to 1. The low voltage detection threshold is determined by the PMC's LVDSC1[LVDV] field. After an LVD reset has occurred, the LVD system holds the MCU in reset until the supply voltage has risen above the low voltage detection threshold. The RCM's SRS0[LVD] bit is set following either an LVD reset or POR.

6.2.2.3 Low leakage wakeup (LLWU)

The LLWU module provides the means for a number of external pins and a number of internal peripherals to wake the MCU from low leakage power modes. The LLWU module is functional only in low leakage power modes. In VLLSx modes, all enabled inputs to the LLWU can generate a system reset.

After a system reset, the LLWU retains the flags indicating the input source of the last wakeup until the user clears them.

NOTE

Some flags are cleared in the LLWU and some flags are required to be cleared in the peripheral module. Refer to the individual peripheral chapters for more information.

6.2.2.4 Multipurpose clock generator loss-of-clock (LOC)

The MCG module supports an external reference clock.

If the MCG_C6[CME] bit in the MCG module is set, the clock monitor is enabled. If the external reference falls below f_{loc_low} or f_{loc_high}, as controlled by the MCG_C2[RANGE] field in the MCG module, the MCU resets. The RCM_SRS0[LOC] bit is set to indicate this reset source.

NOTE

To prevent unexpected loss of clock reset events, all clock monitors must be disabled before entering any low power modes, including VLPR and VLPW.

6.2.2.5 Stop mode acknowledge error (SACKERR)

This reset is generated if the core attempts to enter stop mode or Compute Operation, but not all modules acknowledge stop mode within 1025 cycles of the 1 kHz LPO clock.

A module might not acknowledge the entry to stop mode if an error condition occurs. The error can be caused by a failure of an external clock input to a module.

6.2.2.6 Software reset (SW)

The SYSRESETREQ bit in the NVIC application interrupt and reset control register can be set to force a software reset on the device. (See ARM's NVIC documentation for the full description of the register fields, especially the VECTKEY field requirements.) Setting SYSRESETREQ generates a software reset request. This reset forces a system reset of all major components except for the debug module. A software reset causes the RCM's SRS1[SW] bit to set.

6.2.2.7 Lockup reset (LOCKUP)

The LOCKUP gives immediate indication of seriously errant kernel software. This is the result of the core being locked because of an unrecoverable exception following the activation of the processor's built in system state protection hardware.

The LOCKUP condition causes a system reset and also causes the RCM's SRS1[LOCKUP] bit to set.

6.2.2.8 MDM-AP system reset request

Set the system reset request bit in the MDM-AP control register to initiate a system reset. This is the primary method for resets via the SWD interface. The system reset is held until this bit is cleared.

Set the core hold reset bit in the MDM-AP control register to hold the core in reset as the rest of the chip comes out of system reset.

6.2.2.9 Watchdog timer reset

The watchdog timer (WDOG) monitors the operation of the system by expecting periodic communication from the software. This communication is generally known as servicing (or refreshing) the watchdog. If this periodic refreshing does not occur, the watchdog issues a system reset. The WDOG reset causes RCM_SRS0[WDOG] to set. The watchdog can also be initially tested to verify if it is working and resetting the MCU before deploying for application use. RCM_SRS0[WDOG], along with WDOG_STCTRLH[TESTWDOG], allow the user to identify if the watchdog has forced a reset.

6.2.3 MCU Resets

A variety of resets are generated by the MCU to reset different modules.

6.2.3.1 POR Only

The POR Only reset asserts on the POR reset source only. It resets the PMC.

The POR Only reset also causes all other reset types to occur.

6.2.3.2 Chip POR not VLLS

The Chip POR not VLLS reset asserts on POR and LVD reset sources. It resets parts of the SMC and SIM. It also resets the LPTMR.

The Chip POR not VLLS reset also causes these resets to occur: Chip POR, Chip Reset not VLLS, and Chip Reset (including Early Chip Reset).

6.2.3.3 Chip POR

The Chip POR asserts on POR, LVD, and VLLS Wakeup reset sources. It resets the Reset Pin Filter registers and parts of the SIM and MCG.

The Chip POR also causes the Chip Reset (including Early Chip Reset) to occur.

6.2.3.4 Chip Reset not VLLS

The Chip Reset not VLLS reset asserts on all reset sources except a VLLS Wakeup that does not occur via the $\overline{\text{RESET}}$ pin. It resets parts of the SMC, LLWU, and other modules that remain powered during VLLS mode.

The Chip Reset not VLLS reset also causes the Chip Reset (including Early Chip Reset) to occur.

6.2.3.5 Early Chip Reset

The Early Chip Reset asserts on all reset sources. It resets only the flash memory module. It negates before flash memory initialization begins ("earlier" than when the Chip Reset negates).

6.2.3.6 Chip Reset

Chip Reset asserts on all reset sources and only negates after flash initialization has completed and the $\overline{\text{RESET}}$ pin has also negated. It resets the remaining modules (the modules not reset by other reset types).

6.2.4 Reset Pin

For all reset sources except a VLLS Wakeup that does not occur via the $\overline{\text{RESET}}$ pin, the $\overline{\text{RESET}}$ pin is driven low by the MCU for at least 128 bus clock cycles and until flash initialization has completed.

After flash initialization has completed, the $\overline{\text{RESET}}$ pin is released, and the internal Chip Reset negates after the $\overline{\text{RESET}}$ pin is pulled high. Keeping the $\overline{\text{RESET}}$ pin asserted externally delays the negation of the internal Chip Reset.

The $\overline{\text{RESET}}$ pin can be disabled by programming RESET_PIN_CFG option bit to 0. When this option is selected, there could be a short period of contention during a POR ramp where the device drives the pin out low prior to establishing the setting of this option and releasing the RESET function on the pin.

6.2.5 Debug resets

The following sections detail the debug resets available on the device.

6.2.5.1 Resetting the Debug subsystem

Use the CDBGRSTREQ bit within the DP CTRL/STAT register to reset the debug modules. However, as explained below, using the CDBGRSTREQ bit does not reset all debug-related registers.

CDBGRSTREQ resets the debug-related registers within the following modules:

- SW-DP
- AHB-AP
- MDM-AP (MDM control and status registers)

CDBGRSTREQ does not reset the debug-related registers within the following modules:

- CM0+ core (core debug registers: DHCSR, DCRSR, DCRDR, DEMCR)
- BPU
- DWT
- NVIC
- Crossbar bus switch[1]
- AHB-AP[1]
- Private peripheral bus[1]

1. CDBGRSTREQ does not affect AHB resources so that debug resources on the private peripheral bus are available during System Reset.

6.3 Boot

This section describes the boot sequence, including sources and options.

Some configuration information such as clock trim values stored in factory programmed flash locations is autoloaded.

6.3.1 Boot sources

The CM0+ core adds support for a programmable Vector Table Offset Register (VTOR) to relocate the exception vector table. This device supports booting from internal flash and RAM.

This device supports booting from internal flash with the reset vectors located at addresses 0x0 (initial SP_main), 0x4 (initial PC), and RAM with relocating the exception vector table to RAM.

6.3.2 FOPT boot options

The flash option register (FOPT) in flash memory module (FTFA) allows the user to customize the operation of the MCU at boot time. The register contains read-only bits that are loaded from the NVM's option byte in the flash configuration field. The default setting for all values in the FOPT register is logic 1 since it is copied from the option byte residing in flash, which has all bits as logic 1 in the flash erased state. To configure for alternate settings, program the appropriate bits in the NVM option byte. The new settings will take effect on subsequent POR, VLLSx recoveries, and any system reset. For more details on programming the option byte, refer to the flash memory chapter.

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

Table 6-2. Flash Option Register (FTFA_FOPT) Bit Definitions

Bit Num	Field	Value	Definition
7-6	Reserved	Reserved for future expansion.	
5	FAST_INIT	Select initialization speed on POR, VLLSx, and any system reset .	
		0	Slower initialization. The Flash initialization will be slower with the benefit of reduced average current during this time. The duration of the recovery will be controlled by the clock divider selection determined by the LPBOOT setting.

Table continues on the next page...

Table 6-2. Flash Option Register (FTFA_FOPT) Bit Definitions (continued)

Bit Num	Field	Value	Definition
		1	Fast Initialization.The Flash has faster recoveries at the expense of higher current during these times.
3	RESET_PIN_CFG	Enable/disable control for the RESET pin.	
		0	RESET pin is disabled following a POR and cannot be enabled as RESET function. When this option is selected, there could be a short period of contention during a POR ramp where the device drives the pin out low prior to establishing the setting of this option and releasing the RESET function on the pin. This bit is preserved through system resets and low power modes. When RESET pin function is disabled it cannot be used as a source for low power mode wakeup. **NOTE:** When the reset pin has been disabled and security has been enabled by means of the FSEC register, a mass erase can be performed only by setting both the mass erase and system reset request bits in the MDM-AP register.
		1	RESET pin is dedicated. The port is configured with pullup enabled, open drain, passive filter enabled.
2	NMI_DIS	Enable/disable control for the NMI function.	
		0	NMI interrupts are always blocked. The associated pin continues to default to NMI pin controls with internal pullup enabled. When NMI pin function is disabled it cannot be used as a source for low power mode wakeup.
		1	NMI pin/interrupts reset default to enabled.
1	Reserved	Reserved for future expansion.	
4,0	LPBOOT	Control the reset value of OUTDIV1 value in SIM_CLKDIV1 register. Larger divide value selections produce lower average power consumption during POR, VLLSx recoveries and reset sequencing and after reset exit. The recovery times are also extended if the FAST_INIT option is not selected.	
		00	Core and system clock divider (OUTDIV1) is 0x7 (divide by 8)
		01	Core and system clock divider (OUTDIV1) is 0x3 (divide by 4)
		10	Core and system clock divider (OUTDIV1) is 0x1 (divide by 2)
		11	Core and system clock divider (OUTDIV1) is 0x0 (divide by 1)

6.3.3 Boot sequence

At power up, 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 internally regulated supplies have reached a safe operating voltage as determined by the LVD. The Reset Controller logic then controls a sequence to exit reset.

1. A system reset is held on internal logic, the $\overline{\text{RESET}}$ pin is driven out low, and the MCG is enabled in its default clocking mode.
2. Required clocks are enabled (System Clock, Flash Clock, and any Bus Clocks that do not have clock gate control reset to disabled).

3. The system reset on internal logic continues to be held, but the Flash Controller is released from reset and begins initialization operation while the Reset Control logic continues to drive the RESET pin out low.
4. Early in reset sequencing the NVM option byte is read and stored to the Flash Memory module's FOPT register. If the bits associated with LPBOOT are programmed for an alternate clock divider reset value, the system/core clock is switched to a slower clock speed. If the FAST_INIT bit is programmed clear, the Flash initialization switches to slower clock resulting longer recovery times.
5. When Flash Initialization completes, the RESET pin is released. If RESET continues to be asserted (an indication of a slow rise time on the RESET pin or external drive in low), the system continues to be held in reset. Once the RESET pin is detected high, the Core clock is enabled and the system is released from reset.
6. When the system exits reset, the processor sets up the stack, program counter (PC), and link register (LR). The processor reads the start SP (SP_main) from vector-table offset 0. The core reads the start PC from vector-table offset 4. LR is set to 0xFFFF_FFFF. What happens next depends on the NMI input and the NMI control bit in the Flash Memory module's FOPT register:
    - If the NMI input is high or the NMI function is disabled in the FOPT register, the CPU begins execution at the PC location.
    - If the NMI input is low and the NMI function is enabled in the FOPT register, this results in an NMI interrupt. The processor executes an Exception Entry and reads the NMI interrupt handler address from vector-table offset 8. The CPU begins execution at the NMI interrupt handler.

Subsequent system resets follow this same reset flow.

Chapter 7
Power Management

7.1 Introduction

This chapter describes the various chip power modes and functionality of the individual modules in these modes.

7.2 Clocking Modes

This section describes the various clocking modes supported on this device.

7.2.1 Partial Stop

Partial Stop is a clocking option that can be taken instead of entering Stop mode and is configured in the SMC Stop Control Register (SMC_STOPCTRL). The Stop mode is only partially entered, which leaves some additional functionality alive at the expense of higher power consumption. Partial Stop can be entered from either Run mode or VLP Run mode.

When configured for PSTOP2, only the core and system clocks are gated and the bus clock remains active. The bus masters and bus slaves clocked by the system clock enter Stop mode, but the bus slaves clocked by the bus clock remain in Run (or VLP Run) mode. The clock generators in the MCG and the on-chip regulator in the PMC also remain in Run (or VLP Run) mode. Exit from PSTOP2 can be initiated by a reset, an asynchronous interrupt from a bus master or bus slave clocked by the system clock, or a synchronous interrupt from a bus slave clocked by the bus clock. If configured, a DMA request (using the asynchronous DMA wakeup) can also be used to exit Partial Stop for the duration of a DMA transfer before the device is transitioned back into PSTOP2.

When configured for PSTOP1, both the system clock and the bus clock are gated. All bus masters and bus slaves enter Stop mode, but the clock generators in the MCG and the on-chip regulator in the PMC remain in Run (or VLP Run) mode. Exit from PSTOP1 can be initiated by a reset or an asynchronous interrupt from a bus master or bus slave. If configured, an asynchronous DMA request can also be used to exit Partial Stop for the duration of a DMA transfer before the device is transitioned back into PSTOP1.

PSTOP1 is functionally similar to STOP mode, but offers faster wakeup at the expense of higher power consumption. Another benefit is that it keeps all of the MCG clocks enabled, which can be useful for some of the asynchronous peripherals that can remain functional in Stop modes.

NOTE

If the bus clock is selected as the source for the CLKOUT signal, CLKOUT remains active in PSTOP1 mode.

7.2.2 DMA Wakeup

The DMA can be configured to wake up the device on a DMA request whenever it is placed in Stop mode. The wakeup is configured per DMA channel and is supported in Compute Operation, PSTOP, Stop, and VLPS low power modes.

When a DMA wakeup is detected in PSTOP, Stop or VLPS, then the device initiates a normal exit from the low power mode. This can include restoring the on-chip regulator and internal power switches, enabling the clock generators in the MCG, enabling the system and bus clocks (but not the core clock) and negating the Stop mode signal to the bus masters and bus slaves. The only difference is that the CPU remains in the low power mode with the CPU clock disabled.

During Compute Operation, a DMA wakeup initiates a normal exit from Compute Operation. This includes enabling the clocks and negating the Stop mode signal to the bus masters and bus slaves. The core clock always remains enabled during Compute Operation.

Because the DMA wakeup enables the clocks and negates the Stop mode signals to all bus masters and slaves, software needs to ensure that bus masters and slaves that are not involved with the DMA wakeup and transfer remain in a known state. That can be accomplished by disabling the modules before entry into the low power mode or by setting the Doze enable bit in selected modules.

After the DMA request that initiated the wakeup negates and the DMA completes the current transfer, the device transitions back into the original low power mode. This includes requesting all non-CPU bus masters to enter Stop mode and then requesting bus slaves to enter Stop mode. In Stop and VLPS modes, the MCG and PMC then also enter their appropriate modes.

NOTE

If the requested DMA transfer cannot cause the DMA request to negate, then the device remains in a higher power state until the low power mode is fully exited.

If the DMA request asserts during the Stop mode entry sequence (or reentry if the request asserts during a DMA wakeup), then an enabled DMA wakeup can cause an aborted entry into the low power mode, as well as cause the SMC to assert its Stop Abort flag.

An interrupt that occurs during a DMA wakeup causes an immediate exit from the low power mode (this is optional for Compute Operation) without impacting the DMA transfer.

A DMA wakeup can be generated by either a synchronous DMA request or an asynchronous DMA request. Not all peripherals can generate an asynchronous DMA request in Stop modes. In general, though, if a peripheral can generate synchronous DMA requests and also supports asynchronous interrupts in Stop modes, then it can generate an asynchronous DMA request.

7.2.3 Compute Operation

Compute Operation is an execution or compute-only mode of operation that keeps the CPU enabled with full access to the SRAM and Flash read port, but places all other bus masters and bus slaves into their stop mode. Compute Operation can be enabled in either Run mode or VLP Run mode.

NOTE

Do not enter any stop mode without first exiting Compute Operation.

Because Compute Operation reuses the stop mode logic (including the staged entry with bus masters disabled before bus slaves), any bus master or bus slave that can remain functional in stop mode also remains functional in Compute Operation, including generation of asynchronous interrupts and DMA requests. When enabling Compute Operation in Run mode, module functionality for bus masters and slaves is the equivalent of STOP mode. When enabling Compute Operation in VLP Run mode, module

functionality for bus masters and slaves is the equivalent of VLPS mode. The MCG, PMC, SRAM and Flash read port are not affected by Compute Operation, although the Flash register interface is disabled.

During Compute Operation, the AIPS peripheral space is disabled and attempted accesses generate bus errors. The private peripheral space remains accessible during Compute Operation, including the MCM, NVIC, IOPORT and SysTick. Although access to the GPIO registers via the IOPORT is supported, the GPIO port data input registers do not return valid data since clocks are disabled to the Port Control and Interrupt modules. By writing to the GPIO port data output registers, it is possible to control those GPIO ports that are configured as output pins.

Compute Operation is controlled by the CPO register in the MCM, which is only accessible to the CPU. Setting or clearing the CPOREQ bit in the MCM initiates entry or exit into Compute Operation. Compute Operation can also be configured to exit automatically on detection of an interrupt, which is required in order to service most interrupts. Only the core system interrupts (exceptions, including NMI and SysTick) and any edge sensitive interrupts can be serviced without exiting Compute Operation.

When entering Compute Operation, the CPOACK status bit indicates when entry has completed. When exiting Compute Operation in Run mode, the CPOACK status bit negates immediately. When exiting Compute Operation in VLP Run mode, the exit is delayed to allow the PMC to handle the change in power consumption. This delay means the CPOACK bit is polled to determine when the AIPS peripheral space can be accessed without generating a bus error.

The DMA wakeup is also supported during Compute Operation and causes the CPOACK status bit to clear and the AIPS peripheral space to be accessible for the duration of the DMA wakeup. At the completion of the DMA wakeup, the device transitions back into Compute Operation.

7.2.4 Peripheral Doze

Several peripherals support a peripheral Doze mode, where a register bit can be used to disable the peripheral for the duration of a low power mode. The Flash can also be placed in a low power state during Peripheral Doze via a register bit in the SIM.

Peripheral Doze is defined to include all of the modes of operation listed below.

- The CPU is in wait mode.

- The CPU is in stop mode, including the entry sequence and for the duration of a DMA wakeup.
- The CPU is in Compute Operation, including the entry sequence and for the duration of a DMA wakeup.

Peripheral Doze can therefore be used to disable selected bus masters or slaves for the duration of WAIT or VLPW mode. It can also be used to disable selected bus slaves immediately on entry into any stop mode (or Compute Operation), instead of waiting for the bus masters to acknowledge the entry as part of the stop entry sequence. Finally, it can be used to disable selected bus masters or slaves that should remain inactive during a DMA wakeup.

If the Flash is not being accessed during WAIT and PSTOP modes, then the Flash Doze mode can be used to reduce power consumption, at the expense of a slightly longer wakeup when executing code and vectors from Flash. It can also be used to reduce power consumption during Compute Operation when executing code and vectors from SRAM.

7.2.5 Clock Gating

To conserve power, the clocks to most modules can be turned off using the SCGCx registers in the SIM module. These bits are cleared after any reset, which disables the clock to the corresponding module. Prior to initializing a module, set the corresponding bit in the SCGCx register to enable the clock. Before turning off the clock, make sure to disable the module. For more details, refer to the clock distribution and SIM chapters.

7.3 Power modes

The power management controller (PMC) provides multiple power options to allow the user to optimize power consumption for the level of functionality needed.

Depending on the stop requirements of the user application, a variety of stop modes are available that provide partial power down or full power down of certain logic and/or memory. I/O states are held in all modes of operation. The following table compares the various power modes available.

For each run mode there is a corresponding wait and stop mode. Wait modes are similar to ARM sleep modes. Stop modes (VLPS, STOP) are similar to ARM sleep deep mode. The very low power run (VLPR) operating mode can drastically reduce runtime power when the maximum bus frequency is not required to handle the application needs.

The three primary modes of operation are run, wait and stop. The WFI instruction invokes both wait and stop modes for the chip. The primary modes are augmented in a number of ways to provide lower power based on application needs.

Table 7-1. Chip power modes

Chip mode	Description	Core mode	Normal recovery method
Normal run	Allows maximum performance of chip. Default mode out of reset; on-chip voltage regulator is on.	Run	—
Normal Wait - via WFI	Allows peripherals to function while the core is in sleep mode, reducing power. NVIC remains sensitive to interrupts; peripherals continue to be clocked.	Sleep	Interrupt
Normal Stop - via WFI	Places chip in static state. Lowest power mode that retains all registers while maintaining LVD protection. NVIC is disabled; AWIC is used to wake up from interrupt; peripheral clocks are stopped.	Sleep Deep	Interrupt
VLPR (Very Low Power Run)	On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. Only MCG modes BLPI and BLPE can be used in VLPR. Reduced frequency Flash access mode (1 MHz); LVD off; in BLPI clock mode, only the fast internal reference oscillator is available to provide a low power nominal 4MHz source for the core with the nominal bus and flash clock required to be <800kHz; alternatively, BLPE clock mode can be used with an external clock or the crystal oscillator providing the clock source.	Run	—
VLPW (Very Low Power Wait) -via WFI	Same as VLPR but with the core in sleep mode to further reduce power; NVIC remains sensitive to interrupts (FCLK = ON). On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency.	Sleep	Interrupt
VLPS (Very Low Power Stop)-via WFI	Places chip in static state with LVD operation off. Lowest power mode with ADC and pin interrupts functional. Peripheral clocks are stopped, but OSC, LPTMR, CMP can be used. FTM and UART can optionally be enabled if their clock source is enabled. NVIC is disabled (FCLK = OFF); AWIC is used to wake up from interrupt. On-chip voltage regulator is in a low power mode that supplies only enough power to run the chip at a reduced frequency. All SRAM is operating (content retained and I/O states held).	Sleep Deep	Interrupt
VLLS3 (Very Low Leakage Stop3)	Most peripherals are disabled (with clocks stopped), but OSC, LLWU, LPTMR, CMP can be used. NVIC is disabled; LLWU is used to wake up. SRAM_U and SRAM_L remain powered on (content retained and I/O states held).	Sleep Deep	Wakeup Reset[1]
VLLS1 (Very Low Leakage Stop1)	Most peripherals are disabled (with clocks stopped), but OSC, LLWU, LPTMR, CMP can be used. NVIC is disabled; LLWU is used to wake up. All of SRAM_U and SRAM_L are powered off.	Sleep Deep	Wakeup Reset[1]
VLLS0 (Very Low Leakage Stop 0)	Most peripherals are disabled (with clocks stopped), but LLWU, LPTMR can be used. NVIC is disabled; LLWU is used to wake up. All of SRAM_U and SRAM_L are powered off. LPO disabled, optional POR brown-out detection	Sleep Deep	Wakeup Reset[1]

1. Follows the reset flow with the LLWU interrupt flag set for the NVIC.

7.4 Entering and exiting power modes

The WFI instruction invokes wait and stop modes for the chip. The processor exits the low-power mode via an interrupt. For VLLS modes, the wakeup sources are limited to LLWU generated wakeups, NMI pin, or $\overline{\text{RESET}}$ pin assertions. When the NMI pin or $\overline{\text{RESET}}$ pin have been disabled through associated FOPT settings, then these pins are ignored as wakeup sources. The wake-up flow from VLLSx is always through reset.

NOTE

The WFE instruction can have the side effect of entering a low-power mode, but that is not its intended usage. See ARM documentation for more on the WFE instruction.

On VLLS recoveries, the I/O pins continue to be held in a static state after code execution begins, allowing software to reconfigure the system before unlocking the I/O. RAM is retained in VLLS3 only.

7.5 Module Operation in Low Power Modes

The following table illustrates the functionality of each module while the chip is in each of the low power modes. The standard behavior is shown with some exceptions for Compute Operation (CPO) and Partial Stop2 (PSTOP2).

(Debug modules are discussed separately; see Debug in Low-Power Modes.) Number ratings (such as 4 MHz and 1 Mbit/s) represent the maximum frequencies or maximum data rates per mode. Also, these terms are used:

- FF = Full functionality. In VLPR and VLPW the system frequency is limited, but if a module does not have a limitation in its functionality, it is still listed as FF.
- Async operation = Fully functional with alternate clock source, provided the selected clock source remains enabled
- static = Module register states and associated memories are retained.
- powered = Memory is powered to retain contents.
- low power = Memory is powered to retain contents in a lower power state
- OFF = Modules are powered off; module is in reset state upon wakeup. For clocks, OFF means disabled.
- wakeup = Modules can serve as a wakeup source for the chip.

Table 7-2. Module operation in low power modes

Modules	VLPR	VLPW	Stop	VLPS	VLLSx
Core modules					
NVIC	FF	FF	static	static	OFF
System modules					
Mode Controller	FF	FF	FF	FF	FF
LLWU[1]	static	static	static	static	FF[2]
Regulator	low power	low power	ON	low power	low power in VLLS3, OFF in VLLS0/1
LVD	disabled	disabled	ON	disabled	disabled
Brown-out Detection	ON	ON	ON	ON	ON in VLLS1/3, optionally disabled in VLLS0[3]
DMA	FF Async operation in CPO	FF	Async operation	Async operation	OFF
Watchdog	FF static in CPO	FF	static FF in PSTOP2	static	OFF
EWM	FF static in CPO	FF	static FF in PSTOP2	static	
Clocks					
1kHz LPO	ON	ON	ON	ON	ON in VLLS1/3, OFF in VLLS0
System oscillator (OSC)	OSCERCLK max of 16MHz crystal	OSCERCLK max of 16MHz crystal	OSCERCLK optional	OSCERCLK max of 16MHz crystal	limited to low range/low power in VLLS1/3, OFF in VLLS0
MCG	4 MHz IRC	4 MHz IRC	static - MCGIRCLK optional	static - MCGIRCLK optional	OFF
Core clock	4 MHz max	OFF	OFF	OFF	OFF
Platform clock	4 MHz max	4 MHz max	OFF	OFF	OFF
System clock	4 MHz max OFF in CPO	4 MHz max	OFF	OFF	OFF
Bus clock	1 MHz max OFF in CPO	1 MHz max	OFF 25 MHz max in PSTOP2 from RUN 1 MHz max in PSTOP2 from VLPR	OFF	OFF
Memory and memory interfaces					
Flash	1 MHz max access - no program No register access in CPO	low power	low power	low power	OFF

Table continues on the next page...

Table 7-2. Module operation in low power modes (continued)

Modules	VLPR	VLPW	Stop	VLPS	VLLSx
SRAM	low power	low power	low power	low power	low power in VLLS3, OFF in VLLS0/1
CRC	FF	FF	static	static	OFF
Communication interfaces					
UART0	1 Mbit/s Async operation in CPO	1 Mbit/s	static	static	OFF
UART1	1 Mbit/s Async opearation in CPO	1 Mbit/s	Async operation FF in PSTOP2	Async operation	OFF
SPI0	master mode 500 kbit/s, slave mode 250 kbit/s static, slave mode receive in CPO	master mode 500 kbit/s, slave mode 250 kbit/s	static	static	OFF
I^2C0	50 kbit/s static, address match wakeup in CPO	50 kbit/s	static, address match wakeup FF in PSTOP2	static, address match wakeup	OFF
FlexCAN0	256 kbps	256 kbps	wakeup	wakeup	OFF
Timers					
FTM0, FTM1, and FTM2, FTM3, FTM4, and FTM5	FF Async operation in CPO	FF	static	static	OFF
PDB0 and PDB1	FF Async operation in CPO	FF	static	static	OFF
LPTMR	FF	FF	Async operation FF in PSTOP2	Async operation	Async operation[4]
Analog					
16-bit ADC	FF ADC internal clock only in CPO	FF	ADC internal clock only FF in PSTOP2	ADC internal clock only	OFF
CMP[5]	FF HS or LS compare in CPO	FF	HS or LS compare FF in PSTOP2	HS or LS compare	LS compare in VLLS1/3, OFF in VLLS0
6-bit DAC	FF static in CPO	FF	static FF in PSTOP2	static	static, OFF in VLLS0
12-bit DAC	FF static in CPO	FF	static FF in PSTOP2	static	static
Human-machine interfaces					

Table continues on the next page...

Table 7-2. Module operation in low power modes (continued)

Modules	VLPR	VLPW	Stop	VLPS	VLLSx
GPIO	FF IOPORT write only in CPO	FF	static output, wakeup input FF in PSTOP2	static output, wakeup input	OFF, pins latched

1. Using the LLWU module, the external pins available for this chip do not require the associated peripheral function to be enabled. It only requires the function controlling the pin (GPIO or peripheral) to be configured as an input to allow a transition to occur to the LLWU.
2. Since LPO clock source is disabled, filters will be bypassed during VLLS0.
3. The STOPCTRL[PORPO] bit in the SMC module controls this option.
4. LPO clock source is not available in VLLS0. Also, to use system OSC in VLLS0 it must be configured for bypass (external clock) operation. Pulse counting is available in all modes.
5. CMP in stop or VLPS supports high speed or low speed external pin to pin or external pin to DAC compares. CMP in VLLSx only supports low speed external pin to pin or external pin to DAC compares. Windowed, sampled & filtered modes of operation are not available while in stop, VLPS or VLLSx modes.

Chapter 8
Security

8.1 Introduction

This device implements security based on the mode selected from the flash module. The following sections provide an overview of flash security and details the effects of security on non-flash modules.

8.2 Flash Security

The flash module provides security information to the MCU based on the state held by the FSEC[SEC] bits. The MCU, in turn, confirms the security request and limits access to flash resources. During reset, the flash module initializes the FSEC register using data read from the security byte of the flash configuration field.

NOTE

The security features apply only to external accesses: debug. CPU accesses to the flash are not affected by the status of FSEC.

In the unsecured state all flash commands are available on the programming interfaces either from the debug port (SWD) or user code execution. When the flash is secured (FSEC[SEC] = 00, 01, or 11), the programmer interfaces are only allowed to launch mass erase operations. Additionally, in this mode, the debug port has no access to memory locations.

8.3 Security Interactions with other Modules

The flash security settings are used by the system to determine what resources are available. The following sections describe the interactions between modules and the flash security settings or the impact that the flash security has on non-flash modules.

8.3.1 Security Interactions with Debug

When flash security is active the SWD port cannot access the memory resources of the MCU.

Although most debug functions are disabled, the debugger can write to the Flash Mass Erase in Progress bit to trigger a mass erase (Erase All Blocks) command. A mass erase via the debugger is allowed even when some memory locations are protected.

When mass erase is disabled, mass erase via the debugger is blocked.

Chapter 9
Debug

9.1 Introduction

This device's debug is based on the ARM CoreSight™ architecture and is configured to provide the maximum flexibility as allowed by the restrictions of the pinout and other available resources.

It provides register and memory accessibility from the external debugger interface, basic run/halt control plus two breakpoints and two watchpoints.

Only one debug interface is supported:

- Serial Wire Debug (SWD)
- Micro Trace Buffer (MTB)

9.2 Debug Port Pin Descriptions

The debug port pins default after POR to their SWD functionality.

Table 9-1. Serial wire debug pin description

Pin Name	Type	Description
SWD_CLK	Input	Serial Wire Clock. This pin is the clock for debug logic when in the Serial Wire Debug mode. This pin is pulled down internally.
SWD_DIO	Input / Output	Serial wire debug data input/output. The SWD_DIO pin is used by an external debug tool for communication and device control. This pin is pulled up internally.

9.3 SWD status and control registers

Through the ARM Debug Access Port (DAP), the debugger has access to the status and control elements, implemented as registers on the DAP bus as shown in the following figure. These registers provide additional control and status for low-power mode recovery and typical run-control scenarios. The status register bits also provide a means for the debugger to receive updated status of the core without having to initiate a bus transaction across the crossbar switch, thus remaining less intrusive during a debug session.

It is important to note that these DAP control and status registers are not memory mapped within the system memory map and are only accessible via the Debug Access Port using SWD. The MDM-AP is accessible as Debug Access Port 1 with the available registers shown in the table below.

Table 9-2. MDM-AP Register Summary

Address	Register	Description
0x0100_0000	Status	See MDM-AP Status Register
0x0100_0004	Control	See MDM-AP Control Register
0x0100_00FC	IDR	Read-only identification register that always reads as 0x001C_0020

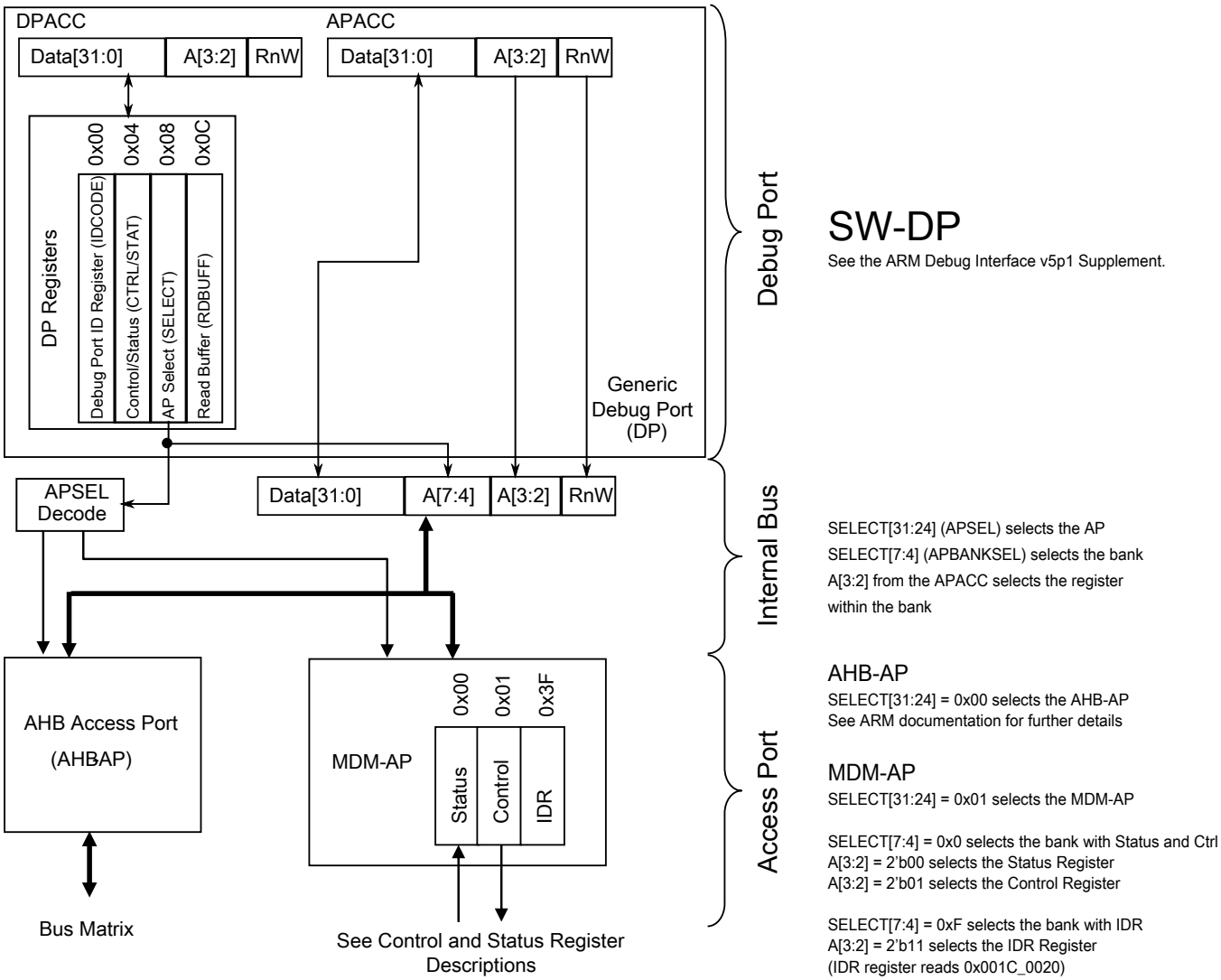


Figure 9-1. MDM AP Addressing

9.3.1 MDM-AP Control Register

Table 9-3. MDM-AP Control register assignments

Bit	Name	Secure[1]	Description
0	Flash Mass Erase in Progress	Y	Set to cause mass erase. Cleared by hardware after mass erase operation completes. When mass erase is disabled (via MEEN), the erase request does not occur and the Flash Mass Erase in Progress bit continues to assert until the next system reset.
1	Debug Disable	N	Set to disable debug. Clear to allow debug operation. When set, it overrides the C_DEBUGEN bit within the DHCSR and force disables Debug logic.
2	Debug Request	N	Set to force the core to halt.

Table continues on the next page...

Table 9-3. MDM-AP Control register assignments (continued)

Bit	Name	Secure[1]	Description
			If the core is in a Stop or Wait mode, this bit can be used to wake up the core and transition to a halted state.
3	System Reset Request	Y	Set to force a system reset. The system remains held in reset until this bit is cleared.
4	Core Hold Reset	N	Configuration bit to control core operation at the end of system reset sequencing. 0 Normal operation—release the core from reset along with the rest of the system at the end of system reset sequencing. 1 Suspend operation—hold the core in reset at the end of reset sequencing. After the system enters this suspended state, clearing this control bit immediately releases the core from reset and CPU operation begins.
5	VLLSx Debug Request (VLLDBGREQ)	N	Set to configure the system to be held in reset after the next recovery from a VLLSx mode. This bit is ignored on a VLLS wakeup via the Reset pin. During a VLLS wakeup via the Reset pin, the system can be held in reset by holding the reset pin asserted, allowing the debugger to reinitialize the debug modules. This bit holds the system in reset when VLLSx modes are exited to allow the debugger time to reinitialize debug IP before the debug session continues. The Mode Controller captures this bit logic on entry to VLLSx modes. On exit from VLLSx modes, the Mode Controller holds the system in reset until VLLDBGACK is asserted. The VLLDBGREQ bit clears automatically due to the POR reset generated as part of the VLLSx recovery.
6	VLLSx Debug Acknowledge (VLLDBGACK)	N	Set to release a system being held in reset following a VLLSx recovery. This bit is used by the debugger to release the system reset when it is being held on VLLSx mode exit. The debugger reinitializes all debug IP and then asserts this control bit to allow the Mode Controller to release the system from reset and allow CPU operation to begin. The VLLDBGACK bit is cleared by the debugger or can be left set because it clears automatically due to the POR reset generated as part of the next VLLSx recovery.
7	VLLSx Status Acknowledge	N	Set this bit to acknowledge the DAP VLLS Status bits have been read. This acknowledge automatically clears the status bits. This bit is used by the debugger to clear the sticky VLLSx mode entry status bits. This bit is asserted and cleared by the debugger.
8 – 31	Reserved for future use	N	

1. Command available in secure mode

9.3.2 MDM-AP Status Register

Table 9-4. MDM-AP Status register assignments

Bit	Name	Description
0	Flash Mass Erase Acknowledge	The Flash Mass Erase Acknowledge bit is cleared after a POR. The bit is also cleared at launch of a mass erase command due to a write of Flash Mass Erase in the Progress bit in the MDM AP Control Register. The Flash Mass Erase Acknowledge is set after flash control logic has started the mass erase operation. When mass erase is disabled (via MEEN), an erase request due to setting of Flash Mass Erase in the Progress bit is not acknowledged.
1	Flash Ready	Indicates flash has been initialized and debugger can be configured even if the system is still held in reset by the debugger.
2	System Security	Indicates the security state. When secure, the debugger does not have access to the system bus or any memory-mapped peripherals. This bit indicates when the part is locked and no system bus access is possible.
3	System Reset	Indicates the system reset state. 0 System is in reset 1 System is not in reset
4	Reserved	
5	Mass Erase Enable	Indicates whether the MCU can be mass-erased or not 0 Mass erase is disabled 1 Mass erase is enabled
6	Backdoor Access Key Enable	Indicates whether the MCU has the backdoor access key enabled. 0 Disabled 1 Enabled
7	LP Enabled	Decode of SMC_PMCTRL[STOPM] field to indicate that VLPS, or VLLSx are the selected power mode the next time the ARM Core enters Deep Sleep. 0 Low Power Stop Mode is not enabled 1 Low Power Stop Mode is enabled Usage intended for debug operation in which Run to VLPS is attempted. Per debug definition, the system actually enters the Stop state. A debugger should interpret deep sleep indication (with SLEEPDEEP and SLEEPING asserted), in conjuntion with this bit asserted as the debugger-VLPS status indication.
8	Very Low Power Mode	Indicates current power mode is VLPx. This bit is not ‘sticky’ and should always represent whether VLPx is enabled. This bit is used to throttle SWD_CLK frequency up/down.
9	Reserved	
10	VLLSx Modes Exit	This bit indicates an exit from VLLSx mode has occurred. The debugger loses communication while the system is in VLLSx (including access to this register). After communication is reestablished, this bit indicates that the system had been in VLLSx. Because the debug modules lose their state during VLLSx modes, they need to be reconfigured.

Table continues on the next page...

Table 9-4. MDM-AP Status register assignments (continued)

Bit	Name	Description
		This bit is set during the VLLSx recovery sequence. The VLLSx Mode Exit bit is held until the debugger has recognized that a VLLS mode was exited. This bit is cleared by a write of 1 to the VLLSx Status Acknowledge bit in the MDM AP Control register.
11 – 15	Reserved for future use	Always read 0.
16	Core Halted	Indicates the Core has entered debug halt mode
17	Core SLEEPDEEP	Indicates the Core has entered a low-power mode
18	Core SLEEPING	SLEEPING==1 and SLEEPDEEP==0 indicates wait or VLPW mode. SLEEPING==1 and SLEEPDEEP==1 indicates stop or VLPS mode.
19 – 31	Reserved for future use	Always read 0.

9.4 Debug Resets

The debug system receives the following sources of reset:

- System POR reset

Conversely, the debug system is capable of generating a system reset using the following mechanisms:

- A system reset in the DAP control register which allows the debugger to hold the system in reset.
- SYSRESETREQ bit in the NVIC application interrupt and reset control register
- A system reset in the DAP control register which allows the debugger to hold the Core in reset.

9.5 Micro Trace Buffer (MTB)

The Micro Trace Buffer (MTB) provides a simple execution trace capability for the Cortex-M0+ processor. When enabled, the MTB records changes in program flow reported by the Cortex-M0+ processor, via the execution trace interface, into a configurable region of the SRAM. Subsequently, an off-chip debugger may extract the trace information, which would allow reconstruction of an instruction flow trace. The MTB does not include any form of load/store data trace capability or tracing of any other information.

In addition to providing the trace capability, the MTB also operates as a simple AHB-Lite SRAM controller. The system bus masters, including the processor, have read/write access to all of the SRAM via the AHB-Lite interface, allowing the memory also to be used to store program and data information. The MTB simultaneously stores the trace information into an attached SRAM and allows bus masters to access the memory. The MTB ensures that trace information write accesses to the SRAM take priority over accesses from the AHB-Lite interface.

The MTB includes trace control registers for configuring and triggering the MTB functions. The MTB also supports triggering via TSTART and TSTOP control functions in the MTB DWT module.

For more details on MTB, see the Micro Trace Buffer (MTB) chapter.

9.6 Debug in Low-Power Modes

In low-power modes in which the debug modules are kept static or powered off, the debugger cannot gather any debug data for the duration of the low-power mode. In the case that the debugger is held static, the debug port returns to full functionality as soon as the low-power mode exits and the system returns to a state with active debug. In the case that the debugger logic is powered off, the debugger is reset on recovery and must be reconfigured after the low-power mode is exited.

Power-mode entry logic monitors Debug Power Up and System Power Up signals from the debug port as indications that a debugger is active. These signals can be changed in RUN, VLPR, WAIT and VLPW. If the debug signal is active and the system attempts to enter Stop or VLPS, FCLK continues to run to support core register access. In these modes in which FCLK is left active the debug modules have access to core registers but not to system memory resources accessed via the crossbar.

With debug enabled, transitions from Run directly to VLPS result in the system entering Stop mode instead. Status bits within the MDM-AP Status register can be evaluated to determine this pseudo-VLPS state. Note with the debug enabled, transitions from Run--> VLPR --> VLPS are still possible.

In VLLS mode, all debug modules are powered off and reset at wakeup.

Going into a VLLSx mode causes all the debug controls and settings to be reset. To give time to the debugger to sync up with the hardware, the MDM-AP Control register can be configured to hold the system in reset on recovery so the debugger can regain control and reconfigure debug logic prior to the system exiting reset and resuming operation.

9.7 Debug & Security

When flash security is enabled (FSEC[SEC] != 10), the debug port capabilities are limited in order to prevent exploitation of secure data. In the secure state the debugger still has access to the MDM-AP Status Register and can determine the current security state of the device. In the case of a secure device that has mass erase disabled (FSEC[MEEN] = 10), attempts to mass erase via the debug interface are blocked. When mass erase is disabled (FSEC[MEEN]= 10), the debugger does not have the capability of performing a mass erase operation via writes to MDM-AP Control Register.

Chapter 10
Signal Multiplexing and Signal Descriptions

10.1 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.

The Port Control block controls which signal is present on the external pin. Reference that chapter to find which register controls the operation of a specific pin.

10.2 Signal Multiplexing Integration

This section summarizes how the module is integrated into the device. For a comprehensive description of the module itself, see the module's dedicated chapter.

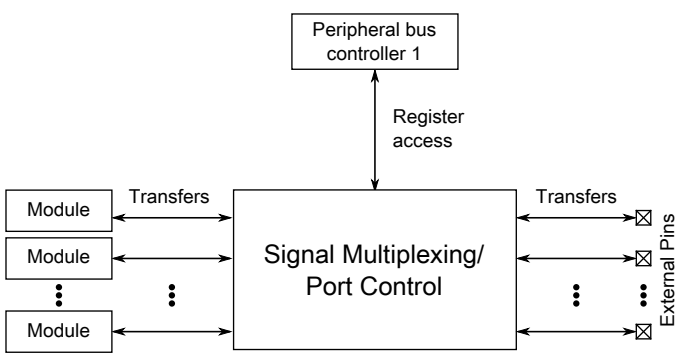


Figure 10-1. Signal multiplexing integration

Table 10-1. Reference links to related information

Topic	Related module	Reference
Full description	Port control	Port control
System memory map		System memory map

Table continues on the next page...

Table 10-1. Reference links to related information (continued)

Topic	Related module	Reference
Clocking		Clock Distribution
Register access	Peripheral bus controller	Peripheral bridge

10.2.1 Port control and interrupt module features

- Five 32-pin ports

NOTE

Not all pins are available on the device. See the following section for details.

- Port A is assigned a dedicated interrupt and Ports B, C, D, and E are grouped to share an interrupt. For DMA requests, Ports A, B, C, D, and E each have a dedicated input to the DMA MUX.

The reset state and read/write characteristics of the bit fields within the PORTx_PCRn registers is summarized in the table below.

Table 10-2. Port control register configuration summary

This field of PORTx_PCRn	Generally resets to	Except for	Resets to	Configurability
PS	1	PTA0	0	Yes - All GPIO are configurable
PE	0	PTA0, PTA3, and PTA4	1	Yes - All GPIO are configurable
DSE	0	No exceptions - all DSE are cleared on reset.	—	8 pins are configurable for High Drive (PTB0 , PTB1, PTC3, PTC4, PTD4, PTD5, PTD6, and PTD7). All are others are fixed for Normal Drive and the associated DSE bit is read only.
SRE	0	No exceptions	—	PTC6 and PTC7 are true open-drain pins, and have no slew rate options.
MUX	000	PTA0, PTA3 and PTA4	111	Yes - All GPIO are configurable

Table continues on the next page...

Table 10-2. Port control register configuration summary (continued)

This field of PORTx_PCRn	Generally resets to	Except for	Resets to	Configurability
PFE	0	No exceptions - all PFE are cleared on reset.[1]	—	The GPIO shared with NMI_b pin is configurable. All other GPIO is fixed and read only.
IRQC	000	No exceptions - all are cleared on reset.	—	Only implemented for ports that support interrupt and DMA functionality.
ISF	0	No exceptions - all are cleared on reset.	—	Only implemented for ports that support interrupt and DMA functionality.

1. The $\overline{\text{RESET}}$ pin has the passive analog filter fixed enabled when functioning as the $\overline{\text{RESET}}$ pin (FOPT[RESET_PIN_CFG] = 1) and fixed disabled when configured for other shared functions.

10.2.2 Clock gating

The clock to the port control module can be gated on and off using SIM_SCGC5[PORTx]. These fields are cleared after any reset, which disables the clock to the corresponding module to conserve power. Prior to initializing the corresponding module, set SIM_SCGC5[PORTx] to enable the clock. Before turning off the clock, make sure to disable the module. For more details, refer to the clock distribution chapter.

10.2.3 Signal multiplexing constraints

1. A given peripheral function must be assigned to a maximum of one package pin. Do not program the same function to more than one pin.
2. To ensure the best signal timing for a given peripheral's interface, choose the pins in closest proximity to each other.

10.3 Pinout

10.3.1 KV11 Signal Multiplexing and Pin Assignments

The following table shows the signals available on each pin and the locations of these pins on the devices supported by this document. The Port Control Module is responsible for selecting which ALT functionality is available on each pin.

NOTE

- PTB0, PTB1, PTC3, PTC4, PTD4, PTD5, PTD6, PTD7 are high current pins.
- PTC6 and PTC7 have open drain outputs

64 LQFP	48 QFP	32 QFN	32 LQFP	Pin Name	DEFAULT	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
—	—	7	7	VDDA/ VREFH	VDDA/ VREFH	VDDA/ VREFH							
—	—	8	8	VREFL/ VSSA	VREFL/ VSSA	VREFL/ VSSA							
1	—	—	—	PTE0	ADC1_SE12	ADC1_SE12	PTE0		UART1_TX				
2	—	—	—	PTE1/ LLWU_P0	ADC1_SE13	ADC1_SE13	PTE1/ LLWU_P0		UART1_RX				
3	1	1	1	VDD	VDD	VDD							
4	2	2	2	VSS	VSS	VSS							
5	3	3	3	PTE16	ADC0_SE1/ ADC0_DP1/ ADC1_SE0	ADC0_SE1/ ADC0_DP1/ ADC1_SE0	PTE16	SPI0_PCS0	UART1_TX	FTM_CLKIN0		FTM0_FLT3	
6	4	4	4	PTE17/ LLWU_P19	ADC0_DM1/ ADC0_SE5/ ADC1_SE5	ADC0_DM1/ ADC0_SE5/ ADC1_SE5	PTE17/ LLWU_P19	SPI0_SCK	UART1_RX	FTM_CLKIN1		LPTMR0_ ALT3	
7	5	5	5	PTE18/ LLWU_P20	ADC0_SE6/ ADC1_SE1/ ADC1_DP1	ADC0_SE6/ ADC1_SE1/ ADC1_DP1	PTE18/ LLWU_P20	SPI0_SOUT	UART1_ CTS_b	I2C0_SDA		SPI0_SIN	
8	6	6	6	PTE19	ADC0_SE7/ ADC1_SE7/ ADC1_DM1	ADC0_SE7/ ADC1_SE7/ ADC1_DM1	PTE19	SPI0_SIN	UART1_ RTS_b	I2C0_SCL		SPI0_SOUT	
9	7	—	—	PTE20	ADC0_SE0/ ADC0_DP0	ADC0_SE0/ ADC0_DP0	PTE20		FTM1_CH0	UART0_TX			
10	8	—	—	PTE21	ADC0_SE4/ ADC0_DM0	ADC0_SE4/ ADC0_DM0	PTE21		FTM1_CH1	UART0_RX			
11	—	—	—	PTE22	ADC0_SE12	ADC0_SE12	PTE22						
12	—	—	—	PTE23	ADC0_SE13	ADC0_SE13	PTE23						
13	9	—	—	VDDA	VDDA	VDDA							
14	10	—	—	VREFH	VREFH	VREFH							
15	11	—	—	VREFL	VREFL	VREFL							
16	12	—	—	VSSA	VSSA	VSSA							
17	13	—	—	PTE29	CMP1_IN5/ CMP0_IN5	CMP1_IN5/ CMP0_IN5	PTE29		FTM0_CH2		FTM_CLKIN0		
18	14	9	9	PTE30	ADC1_SE4/ CMP1_IN4/ DAC0_OUT	ADC1_SE4/ CMP1_IN4/ DAC0_OUT	PTE30		FTM0_CH3		FTM_CLKIN1		
19	—	—	—	PTE31	ADC0_SE14/ CMP0_IN4	ADC0_SE14/ CMP0_IN4	PTE31						
20	15	10	10	PTE24	DISABLED		PTE24	CAN0_TX	FTM0_CH0		I2C0_SCL	EWM_OUT_b	
21	16	11	11	PTE25/ LLWU_P21	DISABLED		PTE25/ LLWU_P21	CAN0_RX	FTM0_CH1		I2C0_SDA	EWM_IN	

64 LQFP	48 QFP	32 QFN	32 LQFP	Pin Name	DEFAULT	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
22	17	12	12	PTA0	SWD_CLK	SWD_CLK	PTA0	UART0_ CTS_b	FTM0_CH5		EWM_IN		SWD_CLK
23	18	13	13	PTA1	DISABLED		PTA1	UART0_RX	FTM2_CH0	CMP0_OUT	FTM2_QD_ PHA	FTM1_CH1	FTM4_CH0
24	19	14	14	PTA2	DISABLED		PTA2	UART0_TX	FTM2_CH1	CMP1_OUT	FTM2_QD_ PHB	FTM1_CH0	FTM4_CH1
25	20	15	15	PTA3	SWD_DIO	SWD_DIO	PTA3	UART0_ RTS_b	FTM0_CH0	FTM2_FLT0	EWM_OUT_b		SWD_DIO
26	21	16	16	PTA4/ LLWU_P3	NMI_b	NMI_b	PTA4/ LLWU_P3		FTM0_CH1	FTM4_FLT0	FTM0_FLT3		NMI_b
27	—	—	—	PTA5	DISABLED		PTA5		FTM0_CH2	FTM5_FLT0			
28	—	—	—	PTA12	DISABLED		PTA12	CAN0_TX	FTM1_CH0				FTM1_QD_ PHA
29	—	—	—	PTA13/ LLWU_P4	DISABLED		PTA13/ LLWU_P4	CAN0_RX	FTM1_CH1				FTM1_QD_ PHB
30	22	—	—	VDD	VDD	VDD							
31	23	—	—	VSS	VSS	VSS							
32	24	17	17	PTA18	EXTAL0	EXTAL0	PTA18		FTM0_FLT2	FTM_CLKIN0		FTM3_CH2	
33	25	18	18	PTA19	XTAL0	XTAL0	PTA19	FTM0_FLT0	FTM1_FLT0	FTM_CLKIN1		LPTMR0_ ALT1	
34	26	19	19	PTA20	RESET_b		PTA20						RESET_b
35	27	20	20	PTB0/ LLWU_P5	ADC0_SE8/ ADC1_SE8	ADC0_SE8/ ADC1_SE8	PTB0/ LLWU_P5	I2C0_SCL	FTM1_CH0			FTM1_QD_ PHA	UART0_RX
36	28	21	21	PTB1	ADC0_SE9/ ADC1_SE9	ADC0_SE9/ ADC1_SE9	PTB1	I2C0_SDA	FTM1_CH1	FTM0_FLT2	EWM_IN	FTM1_QD_ PHB	UART0_TX
37	29	—	—	PTB2	ADC0_SE10/ ADC1_SE10/ ADC1_DM2	ADC0_SE10/ ADC1_SE10/ ADC1_DM2	PTB2	I2C0_SCL	UART0_ RTS_b	FTM0_FLT1		FTM0_FLT3	
38	30	—	—	PTB3	ADC1_SE2/ ADC1_DP2	ADC1_SE2/ ADC1_DP2	PTB3	I2C0_SDA	UART0_ CTS_b			FTM0_FLT0	
39	31	—	—	PTB16	DISABLED		PTB16		UART0_RX	FTM_CLKIN2	CAN0_TX	EWM_IN	
40	32	—	—	PTB17	DISABLED		PTB17		UART0_TX	FTM_CLKIN1	CAN0_RX	EWM_OUT_b	
41	—	—	—	PTB18	DISABLED		PTB18	CAN0_TX		FTM3_CH2			
42	—	—	—	PTB19	DISABLED		PTB19	CAN0_RX		FTM3_CH3			
43	33	—	—	PTC0	ADC1_SE11	ADC1_SE11	PTC0	SPI0_PCS4	PDB_ EXTRG0		CMP0_OUT	FTM0_FLT0	SPI0_PCS0
44	34	22	22	PTC1/ LLWU_P6	ADC1_SE3	ADC1_SE3	PTC1/ LLWU_P6	SPI0_PCS3	UART1_ RTS_b	FTM0_CH0	FTM2_CH0		
45	35	23	23	PTC2	ADC0_SE11/ CMP1_IN0	ADC0_SE11/ CMP1_IN0	PTC2	SPI0_PCS2	UART1_ CTS_b	FTM0_CH1	FTM2_CH1		
46	36	24	24	PTC3/ LLWU_P7	CMP1_IN1	CMP1_IN1	PTC3/ LLWU_P7	SPI0_PCS1	UART1_RX	FTM0_CH2	CLKOUT	FTM3_FLT0	
47	—	—	—	VSS	VSS	VSS							
48	—	—	—	VDD	VDD	VDD							
49	37	25	25	PTC4/ LLWU_P8	DISABLED		PTC4/ LLWU_P8	SPI0_PCS0	UART1_TX	FTM0_CH3		CMP1_OUT	

64 LQFP	48 QFP	32 QFN	32 LQFP	Pin Name	DEFAULT	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
50	38	26	26	PTC5/ LLWU_P9	DISABLED		PTC5/ LLWU_P9	SPI0_SCK	LPTMR0_ ALT2			CMP0_OUT	FTM0_CH2
51	39	27	27	PTC6/ LLWU_P10	CMP0_IN0	CMP0_IN0	PTC6/ LLWU_P10	SPI0_SOUT	PDB_ EXTRG1		UART0_RX		I2C0_SCL
52	40	28	28	PTC7	CMP0_IN1	CMP0_IN1	PTC7	SPI0_SIN			UART0_TX		I2C0_SDA
53	—	—	—	PTC8	ADC1_SE14/ CMP0_IN2	ADC1_SE14/ CMP0_IN2	PTC8		FTM3_CH4				
54	—	—	—	PTC9	ADC1_SE15/ CMP0_IN3	ADC1_SE15/ CMP0_IN3	PTC9		FTM3_CH5				
55	—	—	—	PTC10	ADC1_SE16	ADC1_SE16	PTC10		FTM5_CH0	FTM5_QD_ PHA			
56	—	—	—	PTC11/ LLWU_P11	ADC1_SE17	ADC1_SE17	PTC11/ LLWU_P11		FTM5_CH1	FTM5_QD_ PHB			
57	41	—	—	PTD0/ LLWU_P12	DISABLED		PTD0/ LLWU_P12	SPI0_PCS0	UART0_ CTS_b	FTM0_CH0	UART1_RX	FTM3_CH0	
58	42	—	—	PTD1	ADC0_SE2	ADC0_SE2	PTD1	SPI0_SCK	UART0_ RTS_b	FTM0_CH1	UART1_TX	FTM3_CH1	
59	43	—	—	PTD2/ LLWU_P13	DISABLED		PTD2/ LLWU_P13	SPI0_SOUT	UART0_RX	FTM0_CH2		FTM3_CH2	I2C0_SCL
60	44	—	—	PTD3	DISABLED		PTD3	SPI0_SIN	UART0_TX	FTM0_CH3		FTM3_CH3	I2C0_SDA
61	45	29	29	PTD4/ LLWU_P14	DISABLED		PTD4/ LLWU_P14	SPI0_PCS1	UART0_ RTS_b	FTM0_CH4	FTM2_CH0	EWM_IN	SPI0_PCS0
62	46	30	30	PTD5	ADC0_SE3	ADC0_SE3	PTD5	SPI0_PCS2	UART0_ CTS_b	FTM0_CH5	FTM2_CH1	EWM_OUT_b	SPI0_SCK
63	47	31	31	PTD6/ LLWU_P15	ADC1_SE6	ADC1_SE6	PTD6/ LLWU_P15	FTM4_CH0	UART0_RX	FTM0_CH0	FTM1_CH0	FTM0_FLT0	SPI0_SOUT
64	48	32	32	PTD7	DISABLED		PTD7	FTM4_CH1	UART0_TX	FTM0_CH1	FTM1_CH1	FTM0_FLT1	SPI0_SIN

10.3.2 KV11 Pinouts

The following figure shows the pinout diagram for the devices supported by this document. Many signals may be multiplexed onto a single pin. To determine what signals can be used on which pin, see the previous section.

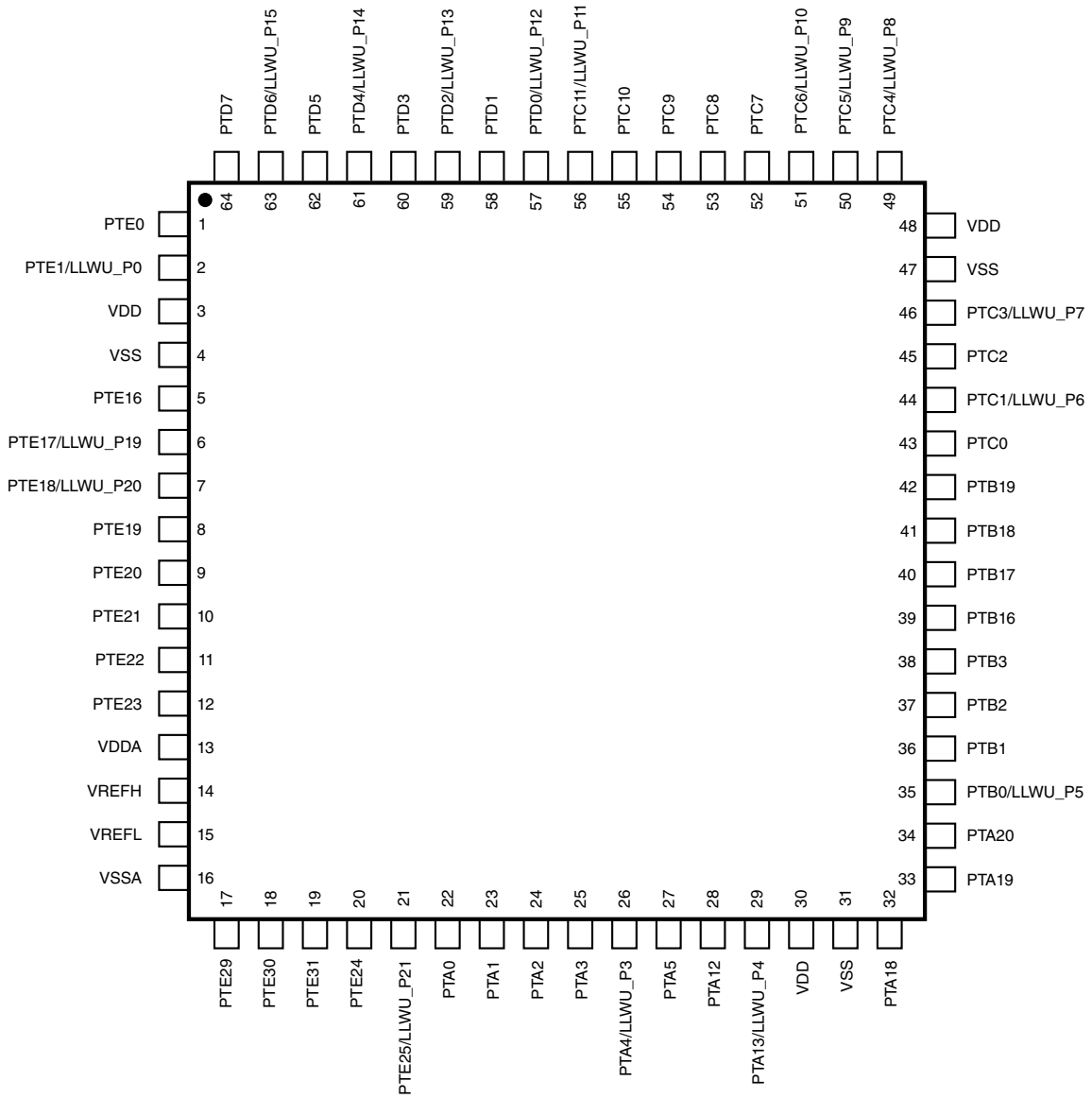


Figure 10-2. 64 LQFP Pinout Diagram

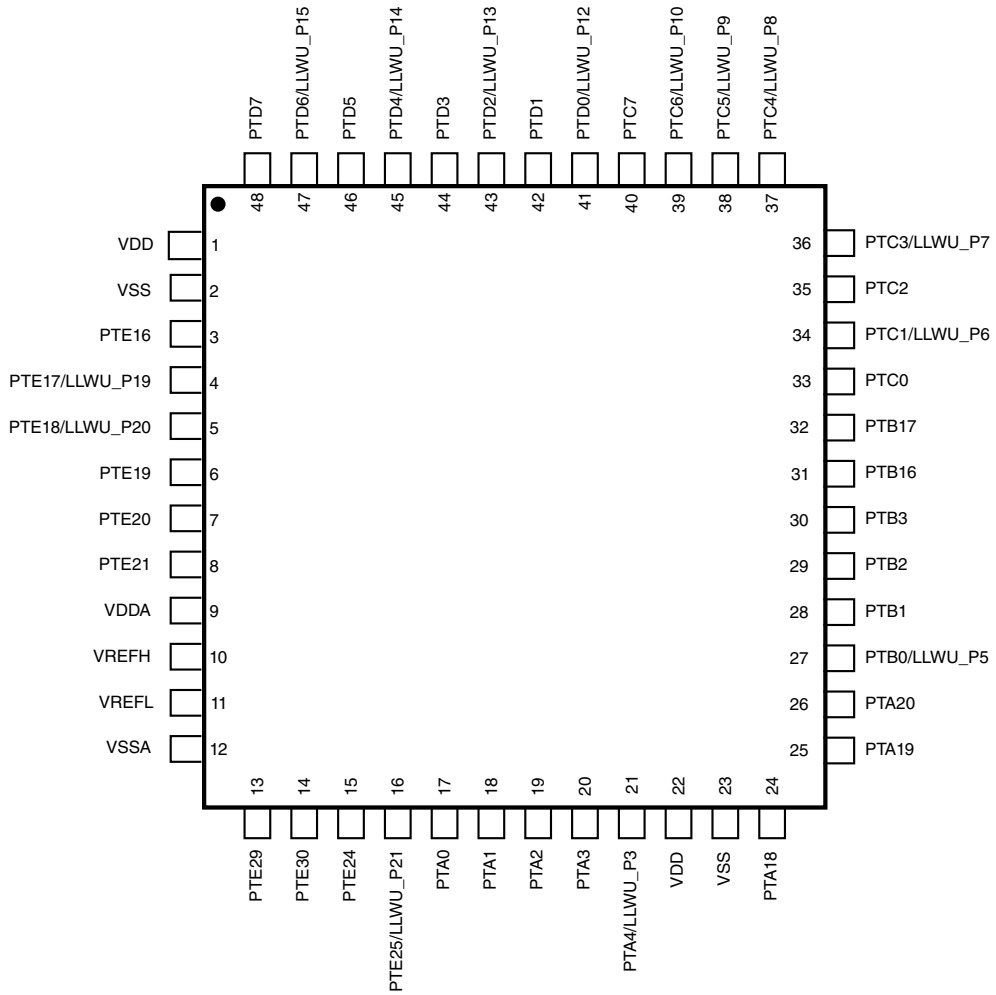


Figure 10-3. 48 QFP Pinout Diagram

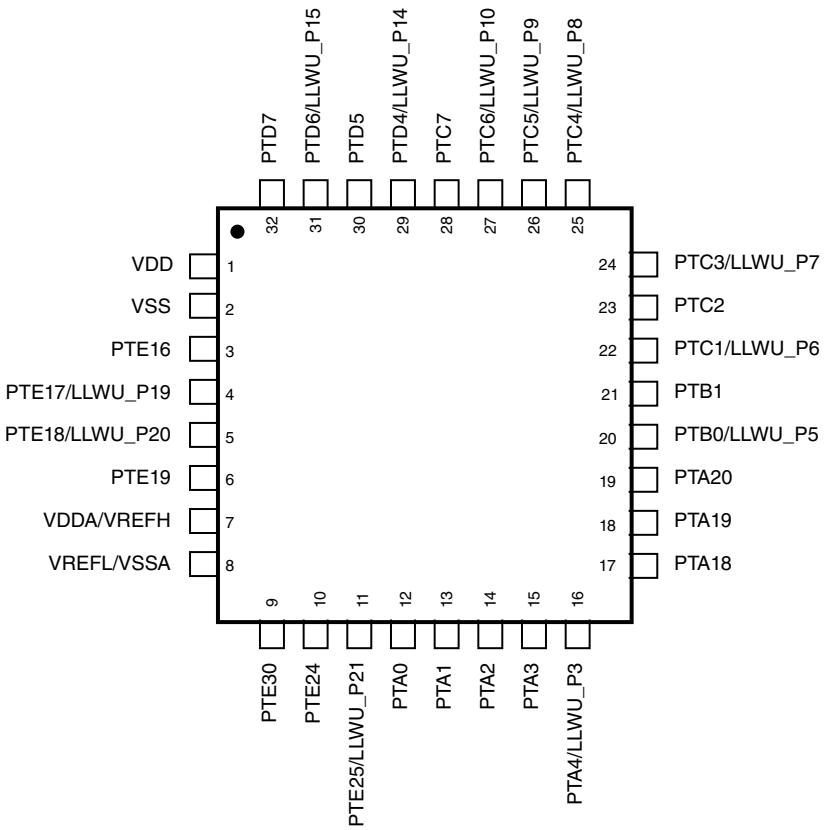


Figure 10-4. 32 LQFP Pinout Diagram

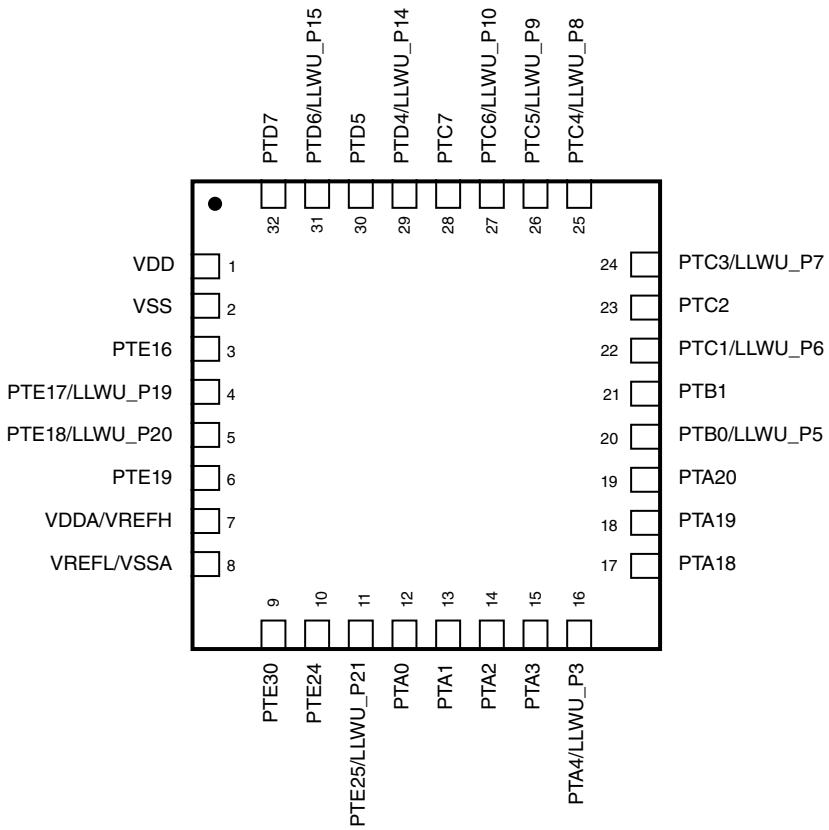


Figure 10-5. 32 QFN Pinout Diagram

10.4 Module Signal Description Tables

The following sections correlate the chip-level signal name with the signal name used in the module's chapter. They also briefly describe the signal function and direction.

10.4.1 Core Modules

Table 10-3. SWD Signal Descriptions

Chip signal name	Module signal name	Description	I/O
SWD_DIO	SWD_DIO	Serial Wire Debug Data Input/Output The SWD_DIO pin is used by an external debug tool for communication and device control. This pin is pulled up internally.	Input / Output
SWD_CLK	SWD_CLK	Serial Wire Clock This pin is the clock for debug logic when in the Serial Wire Debug mode. This pin is pulled down internally.	Input

10.4.2 System Modules

Table 10-4. System Signal Descriptions

Chip signal name	Module signal name	Description	I/O
$\overline{\text{NMI}}$	—	Non-maskable interrupt **NOTE:** Driving the $\overline{\text{NMI}}$ signal low forces a non-maskable interrupt, if the $\overline{\text{NMI}}$ function is selected on the corresponding pin.	I
$\overline{\text{RESET}}$	—	Reset bi-directional signal	I/O
VDD	—	MCU power	I
VSS	—	MCU ground	I

10.4.3 Clock Modules

Table 10-5. OSC Signal Descriptions

Chip signal name	Module signal name	Description	I/O
EXTAL0	EXTAL	External clock/Oscillator input	I
XTAL0	XTAL	Oscillator output	O

10.4.4 Memories and Memory Interfaces

10.4.5 Analog

Table 10-6. ADC0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
ADC0_DP0, ADC0_DP1	DADP3–DADP0	Differential Analog Channel Inputs	I
ADC0_DM0, ADC0_DM1	DADM3–DADM0	Differential Analog Channel Inputs	I
ADC0_SEn	AD*n*	Single-Ended Analog Channel Inputs	I
VREFH	V_{REFSH}	Voltage Reference Select High	I
VREFL	V_{REFSL}	Voltage Reference Select Low	I
VDDA	V_{DDA}	Analog Power Supply	I
VSSA	V_{SSA}	Analog Ground	I

Table 10-7. ADC1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
ADC1_DP1, ADC1_DP2	DADP3–DADP0	Differential Analog Channel Inputs	I
ADC1_DM1, ADC1_DM2	DADM3–DADM0	Differential Analog Channel Inputs	I
ADC1_SEn	AD*n*	Single-Ended Analog Channel Inputs	I
VREFH	V_{REFSH}	Voltage Reference Select High	I
VREFL	V_{REFSL}	Voltage Reference Select Low	I
VDDA	V_{DDA}	Analog Power Supply	I
VSSA	V_{SSA}	Analog Ground	I

Table 10-8. CMP0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
CMP0_IN0, CMP0_IN1, CMP0_IN2, CMP0_IN3, CMP0_IN4, CMP0_IN5	IN[5:0]	Analog voltage inputs	I
CMP0_OUT	CMPO	Comparator output	O

Table 10-9. CMP1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
CMP1_IN0, CMP1_IN1,	IN[5:0]	Analog voltage inputs	I

Table continues on the next page...

Table 10-9. CMP1 Signal Descriptions (continued)

Chip signal name	Module signal name	Description	I/O
CMP1_IN4, CMP0_IN5			
CMP1_OUT	CMPO	Comparator output	O

Table 10-10. DAC0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
DAC0_OUT	—	DAC output	O

10.4.6 Timer Modules

Table 10-11. FTM 0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
FTM_CLKIN[2:0]	EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I
FTM0_CH[5:0]	CHn	FTM channel (n), where n can be 5-0	I/O
FTM0_FLT[3:0]	FAULTj	Fault input (j), where j can be 3-0	I

Table 10-12. FTM 1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
FTM_CLKIN[2:0]	EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I
FTM1_CH[1:0]	CHn	FTM channel (n), where n can be 1-0	I/O
FTM1_FLT[1:0]	FAULTj	Fault input (j), where j can be 1-0	I
FTM1_QD_PHA	PHA	Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A.	I
FTM1_QD_PHB	PHB	Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B.	I

Table 10-13. FTM 2 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
FTM_CLKIN[2:0]	EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I

Table continues on the next page...

Table 10-13. FTM 2 Signal Descriptions (continued)

Chip signal name	Module signal name	Description	I/O
FTM2_CH[1:0]	CHn	FTM channel (n), where n can be 1-0	I/O
FTM2_FLT[1:0]	FAULTj	Fault input (j), where j can be 1-0	I
FTM2_QD_PHA	PHA	Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A.	I
FTM2_QD_PHB	PHB	Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B.	I

Table 10-14. FTM 3 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
FTM_CLKIN[2:0]	EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I
FTM3_CH[5:0]	CHn	FTM channel (n), where n can be 5-0	I/O
FTM3_FLT[3:0]	FAULTj	Fault input (j), where j can be 3-0	I

Table 10-15. FTM 4 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
FTM_CLKIN[2:0]	EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I
FTM4_CH[1:0]	CHn	FTM channel (n), where n can be 1-0	I/O
FTM4_FLT[1:0]	FAULTj	Fault input (j), where j can be 1-0	I
FTM4_CH0	PHA	Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A.	I
FTM4_CH1[1]	PHB	Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B.	I

1. FTM4 signals PHA and PHB signals are connected directly to FTM4_CH0,CH1 pins

Table 10-16. FTM 5 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
FTM_CLKIN[2:0]	EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I
FTM5_CH[1:0]	CHn	FTM channel (n), where n can be 1-0	I/O
FTM5_FLT[1:0]	FAULTj	Fault input (j), where j can be 1-0	I
FTM5_QD_PHA	PHA	Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A.	I
FTM5_QD_PHB	PHB	Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B.	I

Table 10-17. PDB0 and PDB1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
PDB_EXTRG0	EXTRG	External trigger to both PDB0 and PDB1. This signal can trigger either or both PDB0 and PDB1	I
PDB_EXTRG1	EXTRG	PDB_EXTRG0	I

Table 10-18. LPTMR 0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
LPTMR0_ALT[3:1]	LPTMR0_ALT*n*	Pulse Counter Input pin	I

10.4.7 Communication Interfaces

Table 10-19. SPI 0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
SPI0_PCS0	PCS0/$\overline{SS}$	Peripheral Chip Select 0 (O)	I/O
SPI0_PCS[3:1]	PCS[1:3]	Peripheral Chip Selects 1–3	O
SPI0_PCS4	PCS4	Peripheral Chip Select 4	O
SPI0_SIN	SIN	Serial Data In	I
SPI0_SOUT	SOUT	Serial Data Out	O
SPI0_SCK	SCK	Serial Clock (O)	I/O

Table 10-20. I^2C 0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
I2C0_SCL	SCL	Bidirectional serial clock line of the I^2C system.	I/O
I2C0_SDA	SDA	Bidirectional serial data line of the I^2C system.	I/O

Table 10-21. UART 0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
$\overline{UART0\_CTS}$	$\overline{CTS}$	Clear to send	I
$\overline{UART0\_RTS}$	$\overline{RTS}$	Request to send	O
UART0_TX	TXD	Transmit data	O
UART0_RX	RXD	Receive data	I

Table 10-22. UART 1 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
$\overline{\text{UART1\_CTS}}$	$\overline{\text{CTS}}$	Clear to send	I
$\overline{\text{UART1\_RTS}}$	$\overline{\text{RTS}}$	Request to send	O
UART1_TX	TXD	Transmit data	O
UART1_RX	RXD	Receive data	I

Table 10-23. CAN0 Signal Descriptions

Chip signal name	Module signal name	Description	I/O
CAN0_TX	CAN Tx	CAN Transmit Pin	Output
CAN0_RX	CAN Rx	CAN Receive Pin	Input

10.4.8 Human-Machine Interfaces (HMI)

Table 10-24. GPIO Signal Descriptions

Chip signal name	Module signal name	Description	I/O
PTA[31:0][1]	PORTA31–PORTA0	General-purpose input/output	I/O
PTB[31:0][1]	PORTB31–PORTB0	General-purpose input/output	I/O
PTC[31:0][1]	PORTC31–PORTC0	General-purpose input/output	I/O
PTD[31:0][1]	PORTD31–PORTD0	General-purpose input/output	I/O
PTE[31:0][1]	PORTE31–PORTE0	General-purpose input/output	I/O

1. The available GPIO pins depends on the specific package. See the signal multiplexing section for which exact GPIO signals are available.

Chapter 11
Port Control and Interrupts (PORT)

11.1 Introduction

NOTE

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

11.2 Overview

The Port Control and Interrupt (PORT) module provides support for port control, and external interrupt functions.

Most functions can be configured independently for each pin in the 32-bit port and affect the pin regardless of its pin muxing state.

There is one instance of the PORT module for each port. Not all pins within each port are implemented on a specific device.

11.2.1 Features

The PORT module has the following features:

- Pin interrupt on selected pins
  - Interrupt flag and enable registers for each pin
  - Support for edge sensitive (rising, falling, both) or level sensitive (low, high) configured per pin
  - Support for interrupt or DMA request configured per pin
  - Asynchronous wake-up in low-power modes
  - Pin interrupt is functional in all digital pin muxing modes
- Port control

- Individual pull control fields with pullup, pulldown, and pull-disable support on selected pins
- Individual drive strength field supporting high and low drive strength on selected pins
- Individual slew rate field supporting fast and slow slew rates on selected pins
- Individual input passive filter field supporting enable and disable of the individual input passive filter on selected pins
- Individual mux control field supporting analog or pin disabled, GPIO, and up to six chip-specific digital functions
- Pad configuration fields are functional in all digital pin muxing modes.

11.2.2 Modes of operation

11.2.2.1 Run mode

In Run mode, the PORT operates normally.

11.2.2.2 Wait mode

In Wait mode, PORT continues to operate normally and may be configured to exit the Low-Power mode if an enabled interrupt is detected. DMA requests are still generated during the Wait mode, but do not cause an exit from the Low-Power mode.

11.2.2.3 Stop mode

In Stop mode, the PORT can be configured to exit the Low-Power mode via an asynchronous wake-up signal if an enabled interrupt is detected.

11.2.2.4 Debug mode

In Debug mode, PORT operates normally.

11.3 External signal description

The table found here describes the PORT external signal.

Table 11-1. Signal properties

Name	Function	I/O	Reset	Pull
PORTx[31:0]	External interrupt	I/O	0	-

NOTE

Not all pins within each port are implemented on each device.

11.4 Detailed signal description

The table found here contains the detailed signal description for the PORT interface.

Table 11-2. PORT interface—detailed signal description

Signal	I/O	Description	
PORTx[31:0]	I/O	External interrupt.	
		State meaning	Asserted—pin is logic 1. Negated—pin is logic 0.
		Timing	Assertion—may occur at any time and can assert asynchronously to the system clock. Negation—may occur at any time and can assert asynchronously to the system clock.

11.5 Memory map and register definition

Any read or write access to the PORT memory space that is outside the valid memory map results in a bus error. All register accesses complete with zero wait states.

PORT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_9000	Pin Control Register n (PORTA_PCR0)	32	R/W	See section	11.5.1/203
4004_9004	Pin Control Register n (PORTA_PCR1)	32	R/W	See section	11.5.1/203
4004_9008	Pin Control Register n (PORTA_PCR2)	32	R/W	See section	11.5.1/203

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_900C	Pin Control Register n (PORTA_PCR3)	32	R/W	See section	11.5.1/203
4004_9010	Pin Control Register n (PORTA_PCR4)	32	R/W	See section	11.5.1/203
4004_9014	Pin Control Register n (PORTA_PCR5)	32	R/W	See section	11.5.1/203
4004_9018	Pin Control Register n (PORTA_PCR6)	32	R/W	See section	11.5.1/203
4004_901C	Pin Control Register n (PORTA_PCR7)	32	R/W	See section	11.5.1/203
4004_9020	Pin Control Register n (PORTA_PCR8)	32	R/W	See section	11.5.1/203
4004_9024	Pin Control Register n (PORTA_PCR9)	32	R/W	See section	11.5.1/203
4004_9028	Pin Control Register n (PORTA_PCR10)	32	R/W	See section	11.5.1/203
4004_902C	Pin Control Register n (PORTA_PCR11)	32	R/W	See section	11.5.1/203
4004_9030	Pin Control Register n (PORTA_PCR12)	32	R/W	See section	11.5.1/203
4004_9034	Pin Control Register n (PORTA_PCR13)	32	R/W	See section	11.5.1/203
4004_9038	Pin Control Register n (PORTA_PCR14)	32	R/W	See section	11.5.1/203
4004_903C	Pin Control Register n (PORTA_PCR15)	32	R/W	See section	11.5.1/203
4004_9040	Pin Control Register n (PORTA_PCR16)	32	R/W	See section	11.5.1/203
4004_9044	Pin Control Register n (PORTA_PCR17)	32	R/W	See section	11.5.1/203
4004_9048	Pin Control Register n (PORTA_PCR18)	32	R/W	See section	11.5.1/203
4004_904C	Pin Control Register n (PORTA_PCR19)	32	R/W	See section	11.5.1/203
4004_9050	Pin Control Register n (PORTA_PCR20)	32	R/W	See section	11.5.1/203
4004_9054	Pin Control Register n (PORTA_PCR21)	32	R/W	See section	11.5.1/203
4004_9058	Pin Control Register n (PORTA_PCR22)	32	R/W	See section	11.5.1/203
4004_905C	Pin Control Register n (PORTA_PCR23)	32	R/W	See section	11.5.1/203
4004_9060	Pin Control Register n (PORTA_PCR24)	32	R/W	See section	11.5.1/203
4004_9064	Pin Control Register n (PORTA_PCR25)	32	R/W	See section	11.5.1/203
4004_9068	Pin Control Register n (PORTA_PCR26)	32	R/W	See section	11.5.1/203
4004_906C	Pin Control Register n (PORTA_PCR27)	32	R/W	See section	11.5.1/203
4004_9070	Pin Control Register n (PORTA_PCR28)	32	R/W	See section	11.5.1/203
4004_9074	Pin Control Register n (PORTA_PCR29)	32	R/W	See section	11.5.1/203
4004_9078	Pin Control Register n (PORTA_PCR30)	32	R/W	See section	11.5.1/203
4004_907C	Pin Control Register n (PORTA_PCR31)	32	R/W	See section	11.5.1/203
4004_9080	Global Pin Control Low Register (PORTA_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/206
4004_9084	Global Pin Control High Register (PORTA_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/206
4004_90A0	Interrupt Status Flag Register (PORTA_ISFR)	32	w1c	0000_0000h	11.5.4/207
4004_A000	Pin Control Register n (PORTB_PCR0)	32	R/W	See section	11.5.1/203
4004_A004	Pin Control Register n (PORTB_PCR1)	32	R/W	See section	11.5.1/203
4004_A008	Pin Control Register n (PORTB_PCR2)	32	R/W	See section	11.5.1/203

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_A00C	Pin Control Register n (PORTB_PCR3)	32	R/W	See section	11.5.1/203
4004_A010	Pin Control Register n (PORTB_PCR4)	32	R/W	See section	11.5.1/203
4004_A014	Pin Control Register n (PORTB_PCR5)	32	R/W	See section	11.5.1/203
4004_A018	Pin Control Register n (PORTB_PCR6)	32	R/W	See section	11.5.1/203
4004_A01C	Pin Control Register n (PORTB_PCR7)	32	R/W	See section	11.5.1/203
4004_A020	Pin Control Register n (PORTB_PCR8)	32	R/W	See section	11.5.1/203
4004_A024	Pin Control Register n (PORTB_PCR9)	32	R/W	See section	11.5.1/203
4004_A028	Pin Control Register n (PORTB_PCR10)	32	R/W	See section	11.5.1/203
4004_A02C	Pin Control Register n (PORTB_PCR11)	32	R/W	See section	11.5.1/203
4004_A030	Pin Control Register n (PORTB_PCR12)	32	R/W	See section	11.5.1/203
4004_A034	Pin Control Register n (PORTB_PCR13)	32	R/W	See section	11.5.1/203
4004_A038	Pin Control Register n (PORTB_PCR14)	32	R/W	See section	11.5.1/203
4004_A03C	Pin Control Register n (PORTB_PCR15)	32	R/W	See section	11.5.1/203
4004_A040	Pin Control Register n (PORTB_PCR16)	32	R/W	See section	11.5.1/203
4004_A044	Pin Control Register n (PORTB_PCR17)	32	R/W	See section	11.5.1/203
4004_A048	Pin Control Register n (PORTB_PCR18)	32	R/W	See section	11.5.1/203
4004_A04C	Pin Control Register n (PORTB_PCR19)	32	R/W	See section	11.5.1/203
4004_A050	Pin Control Register n (PORTB_PCR20)	32	R/W	See section	11.5.1/203
4004_A054	Pin Control Register n (PORTB_PCR21)	32	R/W	See section	11.5.1/203
4004_A058	Pin Control Register n (PORTB_PCR22)	32	R/W	See section	11.5.1/203
4004_A05C	Pin Control Register n (PORTB_PCR23)	32	R/W	See section	11.5.1/203
4004_A060	Pin Control Register n (PORTB_PCR24)	32	R/W	See section	11.5.1/203
4004_A064	Pin Control Register n (PORTB_PCR25)	32	R/W	See section	11.5.1/203
4004_A068	Pin Control Register n (PORTB_PCR26)	32	R/W	See section	11.5.1/203
4004_A06C	Pin Control Register n (PORTB_PCR27)	32	R/W	See section	11.5.1/203
4004_A070	Pin Control Register n (PORTB_PCR28)	32	R/W	See section	11.5.1/203
4004_A074	Pin Control Register n (PORTB_PCR29)	32	R/W	See section	11.5.1/203
4004_A078	Pin Control Register n (PORTB_PCR30)	32	R/W	See section	11.5.1/203
4004_A07C	Pin Control Register n (PORTB_PCR31)	32	R/W	See section	11.5.1/203
4004_A080	Global Pin Control Low Register (PORTB_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/206
4004_A084	Global Pin Control High Register (PORTB_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/206
4004_A0A0	Interrupt Status Flag Register (PORTB_ISFR)	32	w1c	0000_0000h	11.5.4/207
4004_B000	Pin Control Register n (PORTC_PCR0)	32	R/W	See section	11.5.1/203
4004_B004	Pin Control Register n (PORTC_PCR1)	32	R/W	See section	11.5.1/203
4004_B008	Pin Control Register n (PORTC_PCR2)	32	R/W	See section	11.5.1/203

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_B00C	Pin Control Register n (PORTC_PCR3)	32	R/W	See section	11.5.1/203
4004_B010	Pin Control Register n (PORTC_PCR4)	32	R/W	See section	11.5.1/203
4004_B014	Pin Control Register n (PORTC_PCR5)	32	R/W	See section	11.5.1/203
4004_B018	Pin Control Register n (PORTC_PCR6)	32	R/W	See section	11.5.1/203
4004_B01C	Pin Control Register n (PORTC_PCR7)	32	R/W	See section	11.5.1/203
4004_B020	Pin Control Register n (PORTC_PCR8)	32	R/W	See section	11.5.1/203
4004_B024	Pin Control Register n (PORTC_PCR9)	32	R/W	See section	11.5.1/203
4004_B028	Pin Control Register n (PORTC_PCR10)	32	R/W	See section	11.5.1/203
4004_B02C	Pin Control Register n (PORTC_PCR11)	32	R/W	See section	11.5.1/203
4004_B030	Pin Control Register n (PORTC_PCR12)	32	R/W	See section	11.5.1/203
4004_B034	Pin Control Register n (PORTC_PCR13)	32	R/W	See section	11.5.1/203
4004_B038	Pin Control Register n (PORTC_PCR14)	32	R/W	See section	11.5.1/203
4004_B03C	Pin Control Register n (PORTC_PCR15)	32	R/W	See section	11.5.1/203
4004_B040	Pin Control Register n (PORTC_PCR16)	32	R/W	See section	11.5.1/203
4004_B044	Pin Control Register n (PORTC_PCR17)	32	R/W	See section	11.5.1/203
4004_B048	Pin Control Register n (PORTC_PCR18)	32	R/W	See section	11.5.1/203
4004_B04C	Pin Control Register n (PORTC_PCR19)	32	R/W	See section	11.5.1/203
4004_B050	Pin Control Register n (PORTC_PCR20)	32	R/W	See section	11.5.1/203
4004_B054	Pin Control Register n (PORTC_PCR21)	32	R/W	See section	11.5.1/203
4004_B058	Pin Control Register n (PORTC_PCR22)	32	R/W	See section	11.5.1/203
4004_B05C	Pin Control Register n (PORTC_PCR23)	32	R/W	See section	11.5.1/203
4004_B060	Pin Control Register n (PORTC_PCR24)	32	R/W	See section	11.5.1/203
4004_B064	Pin Control Register n (PORTC_PCR25)	32	R/W	See section	11.5.1/203
4004_B068	Pin Control Register n (PORTC_PCR26)	32	R/W	See section	11.5.1/203
4004_B06C	Pin Control Register n (PORTC_PCR27)	32	R/W	See section	11.5.1/203
4004_B070	Pin Control Register n (PORTC_PCR28)	32	R/W	See section	11.5.1/203
4004_B074	Pin Control Register n (PORTC_PCR29)	32	R/W	See section	11.5.1/203
4004_B078	Pin Control Register n (PORTC_PCR30)	32	R/W	See section	11.5.1/203
4004_B07C	Pin Control Register n (PORTC_PCR31)	32	R/W	See section	11.5.1/203
4004_B080	Global Pin Control Low Register (PORTC_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/206
4004_B084	Global Pin Control High Register (PORTC_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/206
4004_B0A0	Interrupt Status Flag Register (PORTC_ISFR)	32	w1c	0000_0000h	11.5.4/207
4004_C000	Pin Control Register n (PORTD_PCR0)	32	R/W	See section	11.5.1/203
4004_C004	Pin Control Register n (PORTD_PCR1)	32	R/W	See section	11.5.1/203
4004_C008	Pin Control Register n (PORTD_PCR2)	32	R/W	See section	11.5.1/203

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_C00C	Pin Control Register n (PORTD_PCR3)	32	R/W	See section	11.5.1/203
4004_C010	Pin Control Register n (PORTD_PCR4)	32	R/W	See section	11.5.1/203
4004_C014	Pin Control Register n (PORTD_PCR5)	32	R/W	See section	11.5.1/203
4004_C018	Pin Control Register n (PORTD_PCR6)	32	R/W	See section	11.5.1/203
4004_C01C	Pin Control Register n (PORTD_PCR7)	32	R/W	See section	11.5.1/203
4004_C020	Pin Control Register n (PORTD_PCR8)	32	R/W	See section	11.5.1/203
4004_C024	Pin Control Register n (PORTD_PCR9)	32	R/W	See section	11.5.1/203
4004_C028	Pin Control Register n (PORTD_PCR10)	32	R/W	See section	11.5.1/203
4004_C02C	Pin Control Register n (PORTD_PCR11)	32	R/W	See section	11.5.1/203
4004_C030	Pin Control Register n (PORTD_PCR12)	32	R/W	See section	11.5.1/203
4004_C034	Pin Control Register n (PORTD_PCR13)	32	R/W	See section	11.5.1/203
4004_C038	Pin Control Register n (PORTD_PCR14)	32	R/W	See section	11.5.1/203
4004_C03C	Pin Control Register n (PORTD_PCR15)	32	R/W	See section	11.5.1/203
4004_C040	Pin Control Register n (PORTD_PCR16)	32	R/W	See section	11.5.1/203
4004_C044	Pin Control Register n (PORTD_PCR17)	32	R/W	See section	11.5.1/203
4004_C048	Pin Control Register n (PORTD_PCR18)	32	R/W	See section	11.5.1/203
4004_C04C	Pin Control Register n (PORTD_PCR19)	32	R/W	See section	11.5.1/203
4004_C050	Pin Control Register n (PORTD_PCR20)	32	R/W	See section	11.5.1/203
4004_C054	Pin Control Register n (PORTD_PCR21)	32	R/W	See section	11.5.1/203
4004_C058	Pin Control Register n (PORTD_PCR22)	32	R/W	See section	11.5.1/203
4004_C05C	Pin Control Register n (PORTD_PCR23)	32	R/W	See section	11.5.1/203
4004_C060	Pin Control Register n (PORTD_PCR24)	32	R/W	See section	11.5.1/203
4004_C064	Pin Control Register n (PORTD_PCR25)	32	R/W	See section	11.5.1/203
4004_C068	Pin Control Register n (PORTD_PCR26)	32	R/W	See section	11.5.1/203
4004_C06C	Pin Control Register n (PORTD_PCR27)	32	R/W	See section	11.5.1/203
4004_C070	Pin Control Register n (PORTD_PCR28)	32	R/W	See section	11.5.1/203
4004_C074	Pin Control Register n (PORTD_PCR29)	32	R/W	See section	11.5.1/203
4004_C078	Pin Control Register n (PORTD_PCR30)	32	R/W	See section	11.5.1/203
4004_C07C	Pin Control Register n (PORTD_PCR31)	32	R/W	See section	11.5.1/203
4004_C080	Global Pin Control Low Register (PORTD_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/206
4004_C084	Global Pin Control High Register (PORTD_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/206
4004_C0A0	Interrupt Status Flag Register (PORTD_ISFR)	32	w1c	0000_0000h	11.5.4/207
4004_D000	Pin Control Register n (PORTE_PCR0)	32	R/W	See section	11.5.1/203
4004_D004	Pin Control Register n (PORTE_PCR1)	32	R/W	See section	11.5.1/203
4004_D008	Pin Control Register n (PORTE_PCR2)	32	R/W	See section	11.5.1/203

Table continues on the next page...

PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_D00C	Pin Control Register n (PORTE_PCR3)	32	R/W	See section	11.5.1/203
4004_D010	Pin Control Register n (PORTE_PCR4)	32	R/W	See section	11.5.1/203
4004_D014	Pin Control Register n (PORTE_PCR5)	32	R/W	See section	11.5.1/203
4004_D018	Pin Control Register n (PORTE_PCR6)	32	R/W	See section	11.5.1/203
4004_D01C	Pin Control Register n (PORTE_PCR7)	32	R/W	See section	11.5.1/203
4004_D020	Pin Control Register n (PORTE_PCR8)	32	R/W	See section	11.5.1/203
4004_D024	Pin Control Register n (PORTE_PCR9)	32	R/W	See section	11.5.1/203
4004_D028	Pin Control Register n (PORTE_PCR10)	32	R/W	See section	11.5.1/203
4004_D02C	Pin Control Register n (PORTE_PCR11)	32	R/W	See section	11.5.1/203
4004_D030	Pin Control Register n (PORTE_PCR12)	32	R/W	See section	11.5.1/203
4004_D034	Pin Control Register n (PORTE_PCR13)	32	R/W	See section	11.5.1/203
4004_D038	Pin Control Register n (PORTE_PCR14)	32	R/W	See section	11.5.1/203
4004_D03C	Pin Control Register n (PORTE_PCR15)	32	R/W	See section	11.5.1/203
4004_D040	Pin Control Register n (PORTE_PCR16)	32	R/W	See section	11.5.1/203
4004_D044	Pin Control Register n (PORTE_PCR17)	32	R/W	See section	11.5.1/203
4004_D048	Pin Control Register n (PORTE_PCR18)	32	R/W	See section	11.5.1/203
4004_D04C	Pin Control Register n (PORTE_PCR19)	32	R/W	See section	11.5.1/203
4004_D050	Pin Control Register n (PORTE_PCR20)	32	R/W	See section	11.5.1/203
4004_D054	Pin Control Register n (PORTE_PCR21)	32	R/W	See section	11.5.1/203
4004_D058	Pin Control Register n (PORTE_PCR22)	32	R/W	See section	11.5.1/203
4004_D05C	Pin Control Register n (PORTE_PCR23)	32	R/W	See section	11.5.1/203
4004_D060	Pin Control Register n (PORTE_PCR24)	32	R/W	See section	11.5.1/203
4004_D064	Pin Control Register n (PORTE_PCR25)	32	R/W	See section	11.5.1/203
4004_D068	Pin Control Register n (PORTE_PCR26)	32	R/W	See section	11.5.1/203
4004_D06C	Pin Control Register n (PORTE_PCR27)	32	R/W	See section	11.5.1/203
4004_D070	Pin Control Register n (PORTE_PCR28)	32	R/W	See section	11.5.1/203
4004_D074	Pin Control Register n (PORTE_PCR29)	32	R/W	See section	11.5.1/203
4004_D078	Pin Control Register n (PORTE_PCR30)	32	R/W	See section	11.5.1/203
4004_D07C	Pin Control Register n (PORTE_PCR31)	32	R/W	See section	11.5.1/203
4004_D080	Global Pin Control Low Register (PORTE_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/206
4004_D084	Global Pin Control High Register (PORTE_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/206
4004_D0A0	Interrupt Status Flag Register (PORTE_ISFR)	32	w1c	0000_0000h	11.5.4/207

11.5.1 Pin Control Register n (PORTx_PCR*n*)

NOTE

See the Signal Multiplexing and Pin Assignment chapter for the reset value of this device.

See the GPIO Configuration section for details on the available functions for each pin.

Do not modify pin configuration registers associated with pins that are not available in a reduced-pin package offering. Unbonded pins not available in a package are disabled by default to prevent them from consuming power.

Address: Base address + 0h offset + (4d × i), where i=0d to 31d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							ISF	0				IRQC			
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	0					MUX			0	DSE	Reserved	PFE	0	SRE	PE	PS
W																
Reset	0	0	0	0	0	*	*	*	0	*	0	*	0	*	*	*

* Notes:

- MUX field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- DSE field: Varies by port. See the Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- PFE field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- SRE field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- PE field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.
- PS field: Varies by port. See Signal Multiplexing and Signal Descriptions chapter for reset values per port.

PORTx_PCR*n* field descriptions

Field	Description
31–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 ISF	Interrupt Status Flag This field is read-only for pins that do not support interrupt generation. The pin interrupt configuration is valid in all digital pin muxing modes. 0 Configured interrupt is not detected. 1 Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared.
23–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 IRQC	Interrupt Configuration This field is read-only for pins that do not support interrupt generation. The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt/DMA request as follows: 0000 Interrupt Status Flag (ISF) is disabled. 0001 ISF flag and DMA request on rising edge. 0010 ISF flag and DMA request on falling edge. 0011 ISF flag and DMA request on either edge. 0100 Reserved. 0101 Reserved. 0110 Reserved. 0111 Reserved. 1000 ISF flag and Interrupt when logic 0. 1001 ISF flag and Interrupt on rising-edge. 1010 ISF flag and Interrupt on falling-edge. 1011 ISF flag and Interrupt on either edge. 1100 ISF flag and Interrupt when logic 1. 1101 Reserved. 1110 Reserved. 1111 Reserved.
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 MUX	Pin Mux Control Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot. The corresponding pin is configured in the following pin muxing slot as follows: 000 Pin disabled (Alternative 0) (analog). 001 Alternative 1 (GPIO). 010 Alternative 2 (chip-specific). 011 Alternative 3 (chip-specific).

Table continues on the next page...

PORTx_PCR*n* field descriptions (continued)

Field	Description
	100 Alternative 4 (chip-specific). 101 Alternative 5 (chip-specific). 110 Alternative 6 (chip-specific). 111 Alternative 7 (chip-specific).
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 DSE	Drive Strength Enable This field is read-only for pins that do not support a configurable drive strength. Drive strength configuration is valid in all digital pin muxing modes. 0 Low drive strength is configured on the corresponding pin, if pin is configured as a digital output. 1 High drive strength is configured on the corresponding pin, if pin is configured as a digital output.
5 Reserved	This field is reserved.
4 PFE	Passive Filter Enable This field is read-only for pins that do not support a configurable passive input filter. Passive filter configuration is valid in all digital pin muxing modes. 0 Passive input filter is disabled on the corresponding pin. 1 Passive input filter is enabled on the corresponding pin, if the pin is configured as a digital input. Refer to the device data sheet for filter characteristics.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 SRE	Slew Rate Enable This field is read-only for pins that do not support a configurable slew rate. Slew rate configuration is valid in all digital pin muxing modes. 0 Fast slew rate is configured on the corresponding pin, if the pin is configured as a digital output. 1 Slow slew rate is configured on the corresponding pin, if the pin is configured as a digital output.
1 PE	Pull Enable This field is read-only for pins that do not support a configurable pull resistor. Refer to the Chapter of Signal Multiplexing and Signal Descriptions for the pins that support a configurable pull resistor. Pull configuration is valid in all digital pin muxing modes. 0 Internal pullup or pulldown resistor is not enabled on the corresponding pin. 1 Internal pullup or pulldown resistor is enabled on the corresponding pin, if the pin is configured as a digital input.
0 PS	Pull Select This bit is read only for pins that do not support a configurable pull resistor direction. Pull configuration is valid in all digital pin muxing modes. 0 Internal pulldown resistor is enabled on the corresponding pin, if the corresponding PE field is set. 1 Internal pullup resistor is enabled on the corresponding pin, if the corresponding PE field is set.

11.5.2 Global Pin Control Low Register (PORTx_GPCLR)

Only 32-bit writes are supported to this register.

Address: Base address + 80h 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																0															
W	GPWE																GPWD															
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

PORTx_GPCLR field descriptions

Field	Description
31–16 GPWE	Global Pin Write Enable Selects which Pin Control Registers (15 through 0) bits [15:0] update with the value in GPWD. 0 Corresponding Pin Control Register is not updated with the value in GPWD. 1 Corresponding Pin Control Register is updated with the value in GPWD.
GPWD	Global Pin Write Data Write value that is written to all Pin Control Registers bits [15:0] that are selected by GPWE.

11.5.3 Global Pin Control High Register (PORTx_GPCHR)

Only 32-bit writes are supported to this register.

Address: Base address + 84h 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																0															
W	GPWE																GPWD															
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

PORTx_GPCHR field descriptions

Field	Description
31–16 GPWE	Global Pin Write Enable Selects which Pin Control Registers (31 through 16) bits [15:0] update with the value in GPWD. 0 Corresponding Pin Control Register is not updated with the value in GPWD. 1 Corresponding Pin Control Register is updated with the value in GPWD.
GPWD	Global Pin Write Data Write value that is written to all Pin Control Registers bits [15:0] that are selected by GPWE.

11.5.4 Interrupt Status Flag Register (PORTx_ISFR)

The corresponding bit is read only for pins that do not support interrupt generation.

The pin interrupt configuration is valid in all digital pin muxing modes. The Interrupt Status Flag for each pin is also visible in the corresponding Pin Control Register, and each flag can be cleared in either location.

Address: Base address + A0h 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	ISF																															
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

PORTx_ISFR field descriptions

Field	Description
ISF	Interrupt Status Flag Each bit in the field indicates the detection of the configured interrupt of the same number as the field. 0 Configured interrupt is not detected. 1 Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic 1 is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared.

11.6 Functional description

11.6.1 Pin control

Each port pin has a corresponding Pin Control register, PORT_PCRn, associated with it.

The upper half of the Pin Control register configures the pin's capability to either interrupt the CPU or request a DMA transfer, on a rising/falling edge or both edges as well as a logic level occurring on the port pin. It also includes a flag to indicate that an interrupt has occurred.

The lower half of the Pin Control register configures the following functions for each pin within the 32-bit port.

- Pullup or pulldown enable on selected pins
- Drive strength and slew rate configuration on selected pins

- Passive input filter enable on selected pins
- Pin Muxing mode

The functions apply across all digital pin muxing modes and individual peripherals do not override the configuration in the Pin Control register. For example, if an I^2C function is enabled on a pin, that does not override the pullup configuration for that pin.

When the Pin Muxing mode is configured for analog or is disabled, all the digital functions on that pin are disabled. This includes the pullup and pulldown enables, output buffer enable, input buffer enable, and passive filter enable.

The configuration of each Pin Control register is retained when the PORT module is disabled.

Whenever a pin is configured in any digital pin muxing mode, the input buffer for that pin is enabled allowing the pin state to be read via the corresponding GPIO Port Data Input Register (GPIO_PDIR) or allowing a pin interrupt or DMA request to be generated. If a pin is ever floating when its input buffer is enabled, then this can cause an increase in power consumption and must be avoided. A pin can be floating due to an input pin that is not connected or an output pin that has tri-stated (output buffer is disabled).

Enabling the internal pull resistor (or implementing an external pull resistor) will ensure a pin does not float when its input buffer is enabled; note that the internal pull resistor is automatically disabled whenever the output buffer is enabled allowing the Pull Enable bit to remain set. Configuring the Pin Muxing mode to disabled or analog will disable the pin's input buffer and results in the lowest power consumption.

11.6.2 Global pin control

The two global pin control registers allow a single register write to update the lower half of the pin control register on up to 16 pins, all with the same value.

The global pin control registers are designed to enable software to quickly configure multiple pins within the one port for the same peripheral function. However, the interrupt functions cannot be configured using the global pin control registers.

The global pin control registers are write-only registers, that always read as 0.

11.6.3 External interrupts

The external interrupt capability of the PORT module is available in all digital pin muxing modes provided the PORT module is enabled.

Each pin can be individually configured for any of the following external interrupt modes:

- Interrupt disabled, default out of reset
- Active high level sensitive interrupt
- Active low level sensitive interrupt
- Rising edge sensitive interrupt
- Falling edge sensitive interrupt
- Rising and falling edge sensitive interrupt
- Rising edge sensitive DMA request
- Falling edge sensitive DMA request
- Rising and falling edge sensitive DMA request

The interrupt status flag is set when the configured edge or level is detected on the pin . When not in Stop mode, the input is first synchronized to the bus clock to detect the configured level or edge transition.

The PORT module generates a single interrupt that asserts when the interrupt status flag is set for any enabled interrupt for that port. The interrupt negates after the interrupt status flags for all enabled interrupts have been cleared by writing a logic 1 to the ISF flag in either the PORT_ISFR or PORT_PCRn registers.

The PORT module generates a single DMA request that asserts when the interrupt status flag is set for any enabled DMA request in that port. The DMA request negates after the DMA transfer is completed, because that clears the interrupt status flags for all enabled DMA requests.

During Stop mode, the interrupt status flag for any enabled interrupt is asynchronously set if the required level or edge is detected. This also generates an asynchronous wake-up signal to exit the Low-Power mode.

Chapter 12
System Integration Module (SIM)

12.1 Introduction

The system integration module (SIM) provides system control and chip configuration registers.

12.1.1 Features

- Configuration for system clocking
  - System clock divide values
  - Architectural clock gating control
  - ERCLK32K clock selection
- Flash and System RAM size configuration
- Flextimer external clock, hardware trigger, fault source, and input capture selection
- UART receive/transmit source selection and configuration

12.2 Memory map and register definition

The SIM module contains many fields for selecting the clock source and dividers for various module clocks. See the Clock Distribution chapter for more information, including block diagrams and clock definitions.

NOTE

- The SIM_SOPT1 register is located at a different base address than the other SIM registers.
- The SIM regsiters can be written only in the Supervisor mode. In the User mode, write accesses are blocked and result in a bus error.
- To change SIM_FCFG1 setting, it needs to use 32-bit write operation; other size operation will result a bus error.

SIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_7000	System Options Register 1 (SIM_SOPT1)	32	R/W	0000_0000h	12.2.1/213
4004_8004	System Options Register 2 (SIM_SOPT2)	32	R/W	0000_0000h	12.2.2/214
4004_800C	System Options Register 4 (SIM_SOPT4)	32	R/W	0000_0000h	12.2.3/215
4004_8010	System Options Register 5 (SIM_SOPT5)	32	R/W	0000_0000h	12.2.4/218
4004_8014	Systems Option Register 6 (SIM_SOPT6)	32	R/W	0000_0000h	12.2.5/220
4004_8018	System Options Register 7 (SIM_SOPT7)	32	R/W	0000_0000h	12.2.6/222
4004_801C	System Options Register 8 (SIM_SOPT8)	32	R/W	0000_0000h	12.2.7/225
4004_8020	System Options Register 9 (SIM_SOPT9)	32	R/W	0000_0000h	12.2.8/228
4004_8024	System Device Identification Register (SIM_SDID)	32	R	Undefined	12.2.9/229
4004_8034	System Clock Gating Control Register 4 (SIM_SCGC4)	32	R/W	F000_0030h	12.2.10/231
4004_8038	System Clock Gating Control Register 5 (SIM_SCGC5)	32	R/W	0004_0180h	12.2.11/232
4004_803C	System Clock Gating Control Register 6 (SIM_SCGC6)	32	R/W	0000_0001h	12.2.12/234
4004_8040	System Clock Gating Control Register 7 (SIM_SCGC7)	32	R/W	0000_0100h	12.2.13/236
4004_8044	System Clock Divider Register 1 (SIM_CLKDIV1)	32	R/W	See section	12.2.14/237
4004_804C	Flash Configuration Register 1 (SIM_FCFG1)	32	R	See section	12.2.15/239
4004_8050	Flash Configuration Register 2 (SIM_FCFG2)	32	R	See section	12.2.16/240
4004_8058	Unique Identification Register Mid-High (SIM_UIDMH)	32	R	Undefined	12.2.17/241
4004_805C	Unique Identification Register Mid Low (SIM_UIDML)	32	R	Undefined	12.2.18/242
4004_8060	Unique Identification Register Low (SIM_UIDL)	32	R	Undefined	12.2.19/242
4004_8100	WDOG Control Register (SIM_WDOGC)	32	R/W	0000_0000h	12.2.20/243

12.2.1 System Options Register 1 (SIM_SOPT1)

NOTE

The reset value of the SOPT1 register is as follows:

- Exit from POR and LVD: OSC32KSEL is cleared.
- Exit from VLLS or other system reset: OSC32KSEL are unaffected

Address: 4004_7000h base + 0h offset = 4004_7000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												OSC32KSEL		0	
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															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

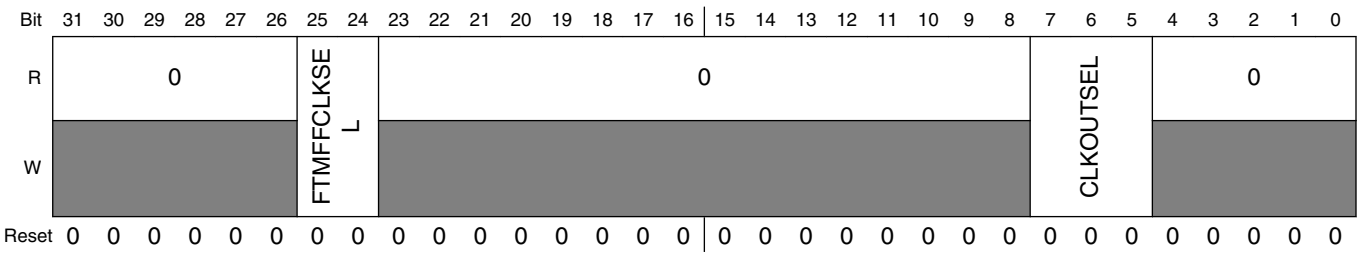
SIM_SOPT1 field descriptions

Field	Description
31–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–18 OSC32KSEL	32K Oscillator Clock Select Selects the 32 kHz clock source (ERCLK32K) for LPTMR. This field is reset only for POR/LVD. 00 System oscillator (OSC32KCLK) 01 Reserved 10 Reserved 11 LPO 1 kHz
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.2 System Options Register 2 (SIM_SOPT2)

SOPT2 contains the controls for selecting many of the module clock source options on this device. See the Clock Distribution chapter for more information including clocking diagrams and definitions of device clocks.

Address: 4004_7000h base + 1004h offset = 4004_8004h

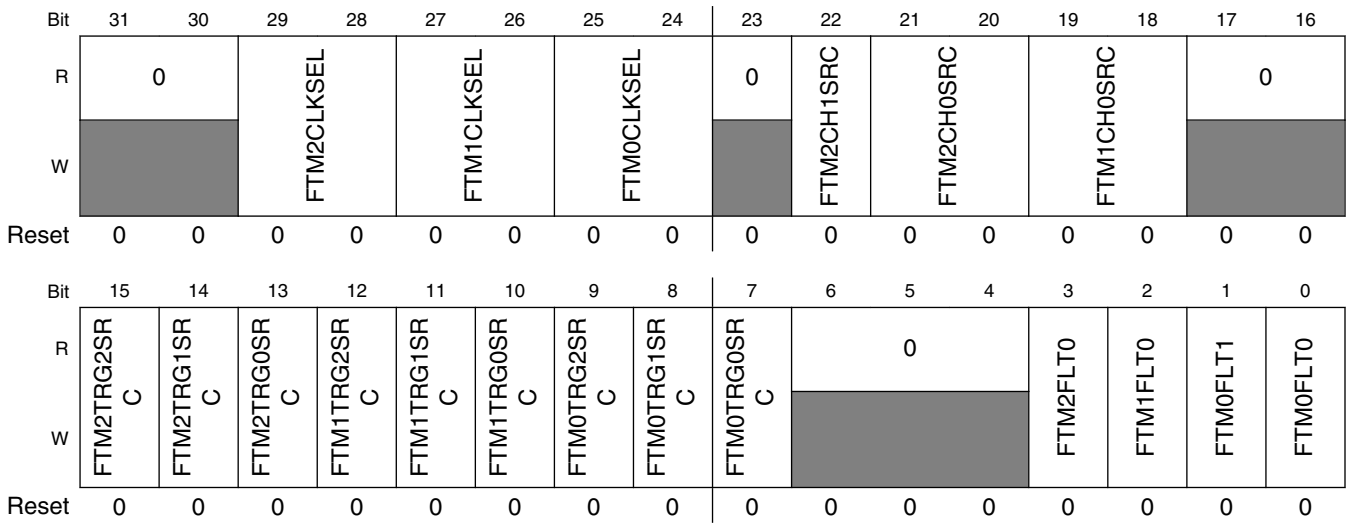


SIM_SOPT2 field descriptions

Field	Description
31–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 FTMFFCLKSEL	FTM Fixed Frequency Clock Select Selects the fixed frequency clock for FTM0, FTM1, FTM2, FTM3, FTM4, and FTM5. 00 MCGFFCLK 01 MCGIRCLK 10 OSCERCLK 11 MCGFFCLK
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–5 CLKOUTSEL	CLKOUT Select Selects the clock to output on the CLKOUT pin. 000 Reserved 001 Reserved 010 Bus clock 011 LPO clock (1 kHz) 100 MCGIRCLK 101 Reserved 110 OSCERCLK 111 Reserved
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.3 System Options Register 4 (SIM_SOPT4)

Address: 4004_7000h base + 100Ch offset = 4004_800Ch



SIM_SOPT4 field descriptions

Field	Description
31–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–28 FTM2CLKSEL	FTM2 External Clock Pin Select Selects the external pin used to drive the clock to the FTM2 module. **NOTE:** The selected pin must also be configured for the FTM external clock function through the appropriate pin control register in the port control module. 00 FTM2 external clock driven by FTM_CLKIN0 pin 01 FTM2 external clock driven by FTM_CLKIN1 pin 10 FTM2 external clock driven by FTM_CLKIN2 pin 11 Reserved
27–26 FTM1CLKSEL	FTM1 External Clock Pin Select Selects the external pin used to drive the clock to the FTM1 module. **NOTE:** The selected pin must also be configured for the FTM external clock function through the appropriate pin control register in the port control module. 00 FTM1 external clock driven by FTM_CLKIN0 pin 01 FTM1 external clock driven by FTM_CLKIN1 pin 10 FTM1 external clock driven by FTM_CLKIN2 pin 11 Reserved
25–24 FTM0CLKSEL	FTM0 External Clock Pin Select Selects the external pin used to drive the clock to the FTM0 module.

Table continues on the next page...

SIM_SOPT4 field descriptions (continued)

Field	Description
	NOTE: The selected pin must also be configured for the FTM external clock function through the appropriate pin control register in the port control module. 00 FTM0 external clock driven by FTM_CLKIN0 pin 01 FTM0 external clock driven by FTM_CLKIN1 pin 10 FTM0 external clock driven by FTM_CLKIN2 pin 11 Reserved
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 FTM2CH1SRC	FTM2 Channel 1 Input Capture Source Select Selects the source for FTM2 channel 1 input capture. **NOTE:** When the FTM is not in Input Capture mode, clear this field. 0 FTM2_CH1 pin is fed to FTM2 CH1 1 FTM2_CH1 pin XOR FTM2_CH0 pin XOR FTM1_CH1 pin is fed to FTM2 CH1 **NOTE:** If this field is set, then the three input pins feed FTM2 channel 1 input capture. In this case, FTM1 channel 1 cannot be used for input capture of FTM1, as it has no pin. FTM1 channel1 can be used for Output Compare mode of FTM1, though without an output.
21–20 FTM2CH0SRC	FTM2 Channel 0 Input Capture Source Select Selects the source for FTM2 channel 0 input capture. **NOTE:** When the FTM is not in Input Capture mode, clear this field. 00 FTM2_CH0 signal 01 CMP0 output 10 CMP1 output 11 Reserved
19–18 FTM1CH0SRC	FTM1 Channel 0 Input Capture Source Select Selects the source for FTM1 channel 0 input capture. **NOTE:** When the FTM is not in Input Capture mode, clear this field. 00 FTM1_CH0 signal 01 CMP0 output 10 CMP1 output 11 Reserved
17–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 FTM2TRG2SRC	FlexTimer 2 Hardware Trigger 2 Source Select Selects the source of FTM2 hardware trigger 2. 0 CMP0 output drives FTM2 hardware trigger 2 1 CMP1 output drives FTM2 hardware trigger 2
14 FTM2TRG1SRC	FlexTimer 2 Hardware Trigger 1 Source Select Selects the source of FTM2 hardware trigger 1.

Table continues on the next page...

SIM_SOPT4 field descriptions (continued)

Field	Description
	0 PDB0 output trigger 1 drives FTM2 hardware trigger 1 1 FTM1 channel match drives FTM2 hardware trigger 1
13 FTM2TRG0SRC	FlexTimer 2 Hardware Trigger 0 Source Select Selects the source of FTM2 hardware trigger 0. 0 CMP0 output drives FTM2 hardware trigger 0 1 FTM0 channel match drives FTM2 hardware trigger 0
12 FTM1TRG2SRC	FlexTimer 1 Hardware Trigger 2 Source Select Selects the source of FTM1 hardware trigger 2. 0 CMP0 output drives FTM1 hardware trigger 2 1 CMP1 output drives FTM1 hardware trigger 2
11 FTM1TRG1SRC	FlexTimer 1 Hardware Trigger 1 Source Select Selects the source of FTM1 hardware trigger 1. 0 PDB0 channel 1 trigger drives FTM1 hardware trigger 1 1 FTM2 channel match drives FTM1 hardware trigger 1
10 FTM1TRG0SRC	FlexTimer 1 Hardware Trigger 0 Source Select Selects the source of FTM1 hardware trigger 0. 0 CMP0 output drives FTM1 hardware trigger 0 1 FTM0 channel match drives FTM1 hardware trigger 0
9 FTM0TRG2SRC	FlexTimer 0 Hardware Trigger 2 Source Select Selects the source of FTM0 hardware trigger 2. 0 CMP0 output drives FTM0 hardware trigger 2 1 CMP1 output drives FTM0 hardware trigger 2
8 FTM0TRG1SRC	FlexTimer 0 Hardware Trigger 1 Source Select Selects the source of FTM0 hardware trigger 1. 0 PDB0 channel 1 trigger drives FTM0 hardware trigger 1 1 FTM2 channel match drives FTM0 hardware trigger 1
7 FTM0TRG0SRC	FlexTimer 0 Hardware Trigger 0 Source Select Selects the source of FTM0 hardware trigger 0. 0 CMP0 output drives FTM0 hardware trigger 0 1 FTM1 channel match drives FTM0 hardware trigger 0
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FTM2FLT0	FTM2 Fault 0 Select Selects the source of FTM2 fault 0. **NOTE:** The pin source for fault 0 must be configured for the FTM module fault function through the appropriate pin control register in the port control module.

Table continues on the next page...

SIM_SOPT4 field descriptions (continued)

Field	Description
	0 FTM2_FLT0 pin 1 CMP0 out
2 FTM1FLT0	FTM1 Fault 0 Select Selects the source of FTM1 fault 0. **NOTE:** The pin source for fault 0 must be configured for the FTM module fault function through the appropriate pin control register in the port control module. 0 FTM1_FLT0 pin 1 CMP0 out
1 FTM0FLT1	FTM0 Fault 1 Select Selects the source of FTM0 fault 1. **NOTE:** The pin source for fault 1 must be configured for the FTM module fault function through the appropriate pin control register in the port control module. 0 FTM0_FLT1 pin 1 CMP1 out
0 FTM0FLT0	FTM0 Fault 0 Select Selects the source of FTM0 fault 0. **NOTE:** The pin source for fault 0 must be configured for the FTM module fault function through the appropriate pin control register in the port control module. 0 FTM0_FLT0 pin 1 CMP0 out

12.2.4 System Options Register 5 (SIM_SOPT5)

Address: 4004_7000h base + 1010h offset = 4004_8010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														UART1ODE	UART0ODE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

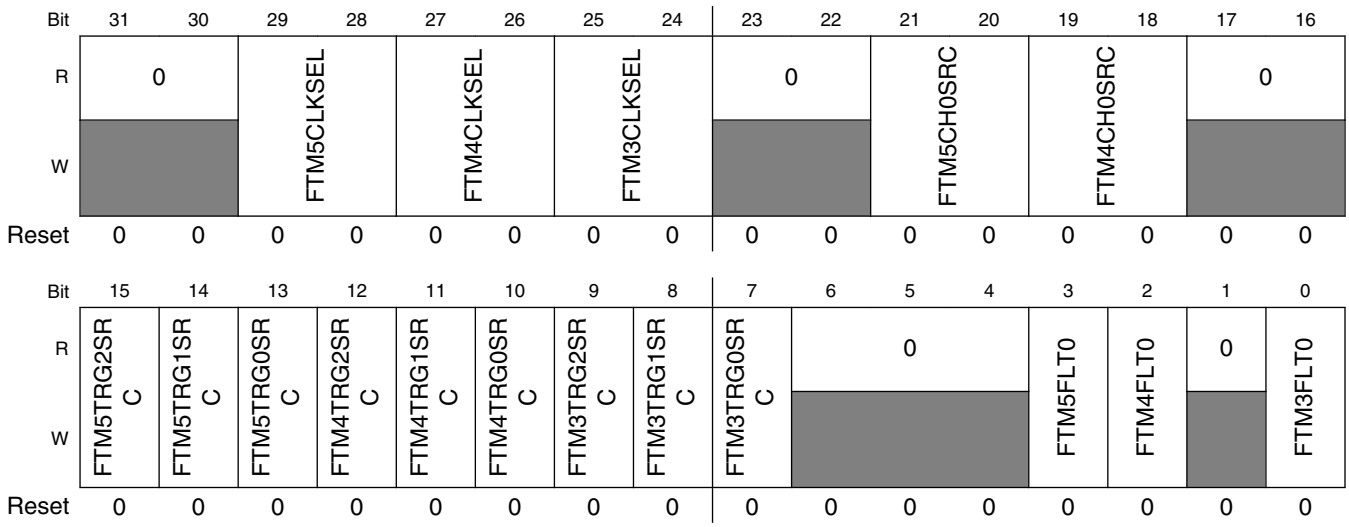
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								UART1RXSRC		UART1TXSRC		UART0RXSRC		UART0TXSRC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SOPT5 field descriptions

Field	Description
31–18 Reserved	This field is reserved.
17 UART1ODE	UART1 Open Drain Enable 0 Open drain is disabled on UART1 1 Open drain is enabled on UART1
16 UART0ODE	UART0 Open Drain Enable 0 Open drain is disabled on UART0 1 Open drain is enabled on UART0
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 UART1RXSRC	UART 1 Receive Data Source Select Selects the source for the UART 1 receive data. 00 UART1_RX pin 01 CMP0 10 CMP1 11 Reserved
5–4 UART1TXSRC	UART 1 Transmit Data Source Select Selects the source for the UART 1 transmit data. 00 UART1_TX pin 01 UART1_TX pin modulated with FTM1 channel 0 output 10 UART1_TX pin modulated with FTM2 channel 0 output 11 Reserved
3–2 UART0RXSRC	UART 0 Receive Data Source Select Selects the source for the UART 0 receive data. 00 UART0_RX pin 01 CMP0 10 CMP1 11 Reserved
UART0TXSRC	UART 0 Transmit Data Source Select Selects the source for the UART 0 transmit data. 00 UART0_TX pin 01 UART0_TX pin modulated with FTM1 channel 0 output 10 UART0_TX pin modulated with FTM2 channel 0 output 11 Reserved

12.2.5 Systems Option Register 6 (SIM_SOPT6)

Address: 4004_7000h base + 1014h offset = 4004_8014h



SIM_SOPT6 field descriptions

Field	Description
31–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–28 FTM5CLKSEL	FTM5 External Clock Pin Select 00 FTM5 external clock driven by FTM_CLKIN0 pin 01 FTM5 external clock driven by FTM_CLKIN1 pin 10 FTM5 external clock driven by FTM_CLKIN2 pin 11 Reserved
27–26 FTM4CLKSEL	FTM4 External Clock Pin Select 00 FTM4 external clock driven by FTM_CLKIN0 pin 01 FTM4 external clock driven by FTM_CLKIN1 pin 10 FTM4 external clock driven by FTM_CLKIN2 pin 11 Reserved
25–24 FTM3CLKSEL	FTM3 External Clock Pin Select 00 FTM3 external clock driven by FTM_CLKIN0 pin 01 FTM3 external clock driven by FTM_CLKIN1 pin 10 FTM3 external clock driven by FTM_CLKIN2 pin 11 Reserved
23–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–20 FTM5CH0SRC	FTM5 channel 0 input capture source select Selects the source for FTM5 channel 0 input capture. 00 FTM5_CH0 signal

Table continues on the next page...

SIM_SOPT6 field descriptions (continued)

Field	Description
	01 CMP0 output 10 CMP1 output 11 Reserved
19–18 FTM4CH0SRC	FTM4 channel 0 input capture source select Selects the source for FTM4 channel 0 input capture. 00 FTM4_CH0 signal 01 CMP0 output 10 CMP1 output 11 Reserved
17–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 FTM5TRG2SRC	FlexTimer 5 Hardware Trigger 2 Source Select 0 CMP0 output drives FTM5 hardware trigger 2 1 CMP1 output drives FTM5 hardware trigger 2
14 FTM5TRG1SRC	FlexTimer 5 Hardware Trigger 1 Source Select 0 PDB1 output trigger 1 drives FTM5 hardware trigger 1 1 FTM4 channel match drives FTM5 hardware trigger 1
13 FTM5TRG0SRC	FlexTimer 5 Hardware Trigger 0 Source Select 0 CMP0 output drives FTM5 hardware trigger 0 1 FTM3 channel match drives FTM5 hardware trigger 0
12 FTM4TRG2SRC	FlexTimer 4 Hardware Trigger 2 Source Select 0 CMP0 output drives FTM4 hardware trigger 2 1 CMP1 output drives FTM4 hardware trigger 2
11 FTM4TRG1SRC	FlexTimer 4 Hardware Trigger 1 Source Select 0 PDB1 output trigger 1 drives FTM4 hardware trigger 1 1 FTM5 channel match drives FTM4 hardware trigger 1
10 FTM4TRG0SRC	FlexTimer 4 Hardware Trigger 0 Source Select 0 CMP0 output drives FTM4 hardware trigger 0 1 FTM3 channel match drives FTM4 hardware trigger 0
9 FTM3TRG2SRC	FlexTimer 3 Hardware Trigger 2 Source Select 0 CMP0 output drives FTM3 hardware trigger 2 1 CMP1 output drives FTM3 hardware trigger 2
8 FTM3TRG1SRC	FlexTimer 3 Hardware Trigger 1 Source Select 0 PDB1 output drives FTM3 hardware trigger 1 1 FTM4 channel match drives FTM3 hardware trigger 1
7 FTM3TRG0SRC	FlexTimer 3 Hardware Trigger 0 Source Select 0 CMP0 output drives FTM3 hardware trigger 0 1 FTM5 channel match drives FTM3 hardware trigger 0

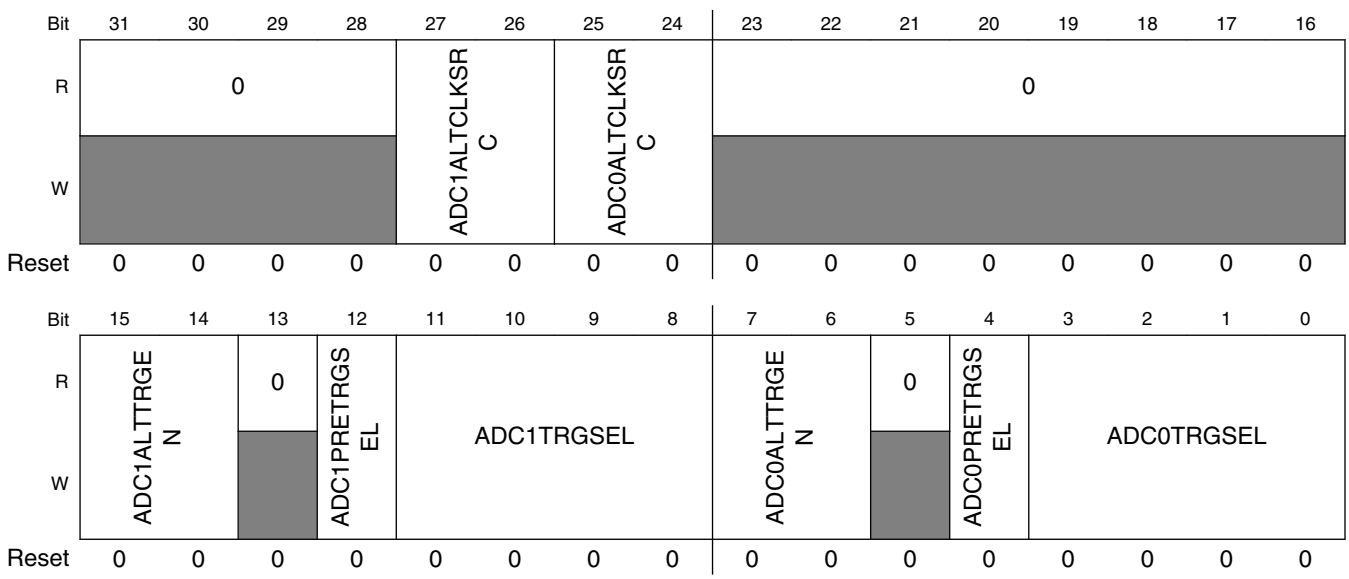
Table continues on the next page...

SIM_SOPT6 field descriptions (continued)

Field	Description
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FTM5FLT0	FTM5 Fault 0 Select 0 FTM5_FLT0 pin 1 CMP0 out
2 FTM4FLT0	FTM4 Fault 0 Select 0 FTM4_FLT0 pin 1 CMP0 out
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 FTM3FLT0	FTM3 Fault 0 Select 0 FTM3_FLT0 pin 1 CMP0 out

12.2.6 System Options Register 7 (SIM_SOPT7)

Address: 4004_7000h base + 1018h offset = 4004_8018h



SIM_SOPT7 field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 ADC1ALTCLKSRC	ADC1 ALT Clock Source Select Selects the source for the ADC1 ALT clock.

Table continues on the next page...

SIM_SOPT7 field descriptions (continued)

Field	Description
	00 OUTDIV5 output 01 MCGIRCLK 10 OSCERCLK 11 Reserved
25–24 ADC0ALTCLKSRC	ADC0 ALT Clock Source Select Selects the source for the ADC0 ALT clock. 00 OUTDIV5 output 01 MCGIRCLK 10 OSCERCLK 11 Reserved
23–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–14 ADC1ALTTRGEN	Enable alternative conversion triggers for ADC1. 00 PDB0 CH1 triggers ADC1 01 PDB1 CH1 triggers ADC1 10 Alt trigger source ADC1TRGSEL 11 PDB0 CH1 OR PDB1 CH1 trigger ADC1
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 ADC1PRETRGSEL	ADC1 Pre-trigger Select Selects the ADC1 pre-trigger source when alternative triggers are enabled through ADC1ALTTRGEN. 0 Pre-trigger A for ADC1. Clearing this field will result in ADHWTSA=1 and ADHWTSB=0. 1 Pre-trigger B for ADC1. Setting this bit will result in ADHWTSA=0 and ADHWTSB=1.
11–8 ADC1TRGSEL	ADC1 Trigger Select Selects the ADC1 trigger source when alternative triggers are functional in Stop and VLPS modes. 0000 External trigger pin input (PDB_EXTRG0) 0001 CMP0 output 0010 CMP1 output 0011 External trigger pin input (PDB_EXTRG1) 0100 DMA channel 0 transfer last write complete 0101 DMA channel 1 transfer last write complete 0110 DMA channel 2 transfer last write complete 0111 DMA channel 3 transfer last write complete 1000 FTM0 intialtrig or external trig output 1001 FTM1 intial trig or external trig output 1010 FTM2 intial trig or external trig output 1011 FTM3 intial trig or external trig output 1100 FTM4 intial trig or external trig output 1101 FTM5 intial trig or external trig output 1110 LPTMR0 trigger 1111 Reserved

Table continues on the next page...

SIM_SOPT7 field descriptions (continued)

Field	Description
7–6 ADC0ALTTRGEN	Enable alternative conversion triggers for ADC0. 00 PDB0 CH0 triggers ADC0 01 PDB1 CH0 triggers ADC0 10 Alt trigger source as per ADC0TRGSEL 11 PDB0 CH0 OR PDB1 CH0 trigger ADC0
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ADC0PRETRGSEL	ADC0 Pre-trigger Select Selects the ADC0 pre-trigger source when alternative triggers are enabled through ADC0ALTTRGEN. 0 Pre-trigger A for ADC0. Clearing this field will result in ADHWTSA=1 and ADHWTSB=0. 1 Pre-trigger B for ADC0. Setting this bit will result in ADHWTSA=0 and ADHWTSB=1.
ADC0TRGSEL	ADC0 Trigger Select Selects the ADC0 trigger source when alternative triggers are functional in Stop and VLPS modes. . 0000 External trigger pin input (PDB_EXTRG0) 0001 CMP0 output 0010 CMP1 output 0011 External trigger pin input (PDB_EXTRG1) 0100 DMA channel 0 transfer last write complete 0101 DMA channel 1 transfer last write complete 0110 DMA channel 2 transfer last write complete 0111 DMA channel 3 transfer last write complete 1000 FTM0 intialtrig or external trig output 1001 FTM1 intial trig or external trig output 1010 FTM2 intial trig or external trig output 1011 FTM3 intial trig or external trig output 1100 FTM4 intial trig or external trig output 1101 FTM5 intial trig or external trig output 1110 LPTMR0 trigger 1111 Reserved

12.2.7 System Options Register 8 (SIM_SOPT8)

NOTE

SOPT8[FTMxSYNCBIT] provides a mechanism to allow two or more FTMs to be initialized at the same time. Each of the FTMs needs FTMx_SYNC[TRIG0] to be set before software synchronization can occur. The user can perform a simultaneous write of logic “1” to SOPT8[FTMxSYNCBIT], which sets FTMx_SYNC[TRIG0] input to the corresponding FTM on the same bus clock cycle . The TRIGx inputs react to a rising edge and the user must clear SOPT8[FTMxSYNCBIT] to allow other hardware trigger inputs to force a synchronization/ refresh event via FTMx_SYNC[TRIG0] input.

Address: 4004_7000h base + 101Ch offset = 4004_801Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								FTM2OCH1SRC	FTM2OCH0SRC	FTM0OCH5SRC	FTM0OCH4SRC	FTM0OCH3SRC	FTM0OCH2SRC	FTM0OCH1SRC	FTM0OCH0SRC
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						CARRIER_SELECT0		0		FTM5SYNCBIT	FMT4SYNCBIT	FMT3SYNCBIT	FTM2SYNCBIT	FTM1SYNCBIT	FTM0SYNCBIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SOPT8 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FTM2OCH1SRC	FTM2 channel 1 output PWM/OCMP mode source select Selects the source for FTM2 channel 1 PWM/OCMP mode. **NOTE:** When the FTM is not in output PWM/OCMP mode, clear this field. 0 FTM2CH1 pin is the output of FTM2 channel 1 PWM/OCMP 1 FTM2CH1 pin is the output of FTM2 channel 1 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT0

Table continues on the next page...

SIM_SOPT8 field descriptions (continued)

Field	Description
22 FTM2OCH0SRC	FTM2 channel 0 output PWM/OCMP mode source select Selects the source for FTM2 channel 0 PWM/OCMP mode. **NOTE:** When the FTM is not in output PWM/OCMP mode, clear this field. 0 FTM2CH0 pin is the output of FTM2 channel 0 PWM/OCMP 1 FTM2CH0 pin is the output of FTM2 channel 0 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT0
21 FTM0OCH5SRC	FTM0 channel 5 output PWM/OCMP mode source select Selects the source for FTM0 channel 5 PWM/OCMP mode. **NOTE:** When the FTM is not in output PWM/OCMP mode, clear this field. 0 FTM0CH5 pin is the output of FTM0 channel 5 PWM/OCMP 1 FTM0CH5 pin is the output of FTM0 channel 5 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT0
20 FTM0OCH4SRC	FTM0 channel 4 output PWM/OCMP mode source select Selects the source for FTM0 channel 4 PWM/OCMP mode. **NOTE:** When the FTM is not in output PWM/OCMP mode, clear this field. 0 FTM0CH4 pin is the output of FTM0 channel 4 PWM/OCMP 1 FTM0CH4 pin is the output of FTM0 channel 4 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT0
19 FTM0OCH3SRC	FTM0 channel 3 output PWM/OCMP mode source select Selects the source for FTM0 channel 3 PWM/OCMP mode. **NOTE:** When the FTM is not in output PWM/OCMP mode, clear this field. 0 FTM0CH3 pin is the output of FTM0 channel 3 PWM/OCMP 1 FTM0CH3 pin is the output of FTM0 channel 3 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT0
18 FTM0OCH2SRC	FTM0 channel 2 output PWM/OCMP mode source select Selects the source for FTM0 channel 2 PWM/OCMP mode. **NOTE:** When the FTM is not in output PWM/OCMP mode, clear this field. 0 FTM0CH2 pin is the output of FTM0 channel 2 PWM/OCMP 1 FTM0CH2 pin is the output of FTM0 channel 2 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT0
17 FTM0OCH1SRC	FTM0 channel 1 output PWM/OCMP mode source select Selects the source for FTM0 channel 1 PWM/OCMP mode. **NOTE:** When the FTM is not in output PWM/OCMP mode, clear this field. 0 FTM0CH1 pin is the output of FTM0 channel 1 PWM/OCMP 1 FTM0CH1 pin is the output of FTM0 channel 1 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT0

Table continues on the next page...

SIM_SOPT8 field descriptions (continued)

Field	Description
16 FTM0OCH0SRC	FTM0 channel 0 output PWM/OCMP mode source select Selects the source for FTM0 channel 0 PWM/OCMP mode. **NOTE:** When the FTM is not in output PWM/OCMP mode, clear this field. 0 FTM0CH0 pin is the output of FTM0 channel 0 PWM/OCMP 1 FTM0CH0 pin is the output of FTM0 channel 0 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT0
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 CARRIER_ SELECT0	Carrier frequency selection for FTM0/2 output channel 00 FTM1_CH1 output provides the carrier signal for Timer Modulation mode 01 LPTMR0 prescaler output provides the carrier signal for Timer Modulation mode 10 FTM5_CH1 output provides the carrier signal for Timer Modulation mode 11 Reserved
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 FTM5SYNCBIT	FlexTimer 5 Hardware Trigger 0 configure by software 0 No effect on FTM5; this allows the hardware trigger options to function as expected. See SOPT6[FTM5TRG0SRC]. 1 If TRIG0 is enabled, this refreshes the FTM5CNTIN and allbuffered registers of the FTM5 (must write 0 first then write 1); this masks the hardware trigger.
4 FMT4SYNCBIT	FlexTimer 4 Hardware Trigger 0 configure by software 0 No effect to FTM4; this allows the hardware trigger options to function as expected. See SOPT6[FTM4TRG0SRC]. 1 If TRIG0 is enabled, this refreshes the FTM4CNTIN and allbuffered registers of the FTM4 (must write 0 first then write 1); this masks the hardware trigger.
3 FMT3SYNCBIT	FlexTimer 3 Hardware Trigger 0 configure by software 0 No effect to FTM3; this allows the hardware trigger options to function as expected. See SOPT6[FTM3TRG0SRC]. 1 If TRIG0 is enabled, this refreshes the FTM3CNTIN and allbuffered registers of the FTM3 (must write 0 first then write 1); this masks the hardware trigger.
2 FTM2SYNCBIT	FlexTimer 2 Hardware Trigger 0 configure by software 0 No effect to FTM2; this allows the hardware trigger options to function as expected. See SOPT4[FTM2TRG0SRC]. 1 If TRIG0 is enabled, this refreshes the FTM2CNTIN and all buffered registers of the FTM2 (must write 0 first then write 1); this masks the hardware trigger.
1 FTM1SYNCBIT	FlexTimer 1 Hardware Trigger 0 configure by software 0 No effect to FTM1; this allows the hardware trigger options to function as expected. See SOPT4[FTM1TRG0SRC]. 1 If TRIG0 enabled, this refreshes the FTM1CNTIN and all buffered registers of the FTM1 (must write 0 first then write 1); this masks the hardware trigger.
0 FTM0SYNCBIT	FlexTimer 0 Hardware Trigger 0 configure by software

Table continues on the next page...

SIM_SOPT8 field descriptions (continued)

Field	Description
	0 No effect to FTM0; this allow the hardware trigger options to function as expected. See SOPT4[FTM0TRG0SRC]. 1 If TRIG0 enabled, this refreshes the FTM0CNTIN and all buffered registers of the FTM0 (must write 0 first then write 1); this masks the hardware trigger.

12.2.8 System Options Register 9 (SIM_SOPT9)

Address: 4004_7000h base + 1020h offset = 4004_8020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								FTM4OCH1SRC	FTM4OCH0SRC	FTM3OCH5SRC	FTM3OCH4SRC	FTM3OCH3SRC	FTM3OCH2SRC	FTM3OCH1SRC	FTM3OCH0SRC
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						CARRIER_SELECT1		0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SOPT9 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FTM4OCH1SRC	FTM4 channel 1 output PWM/OCMP mode source select 0 FTM4CH1 pin is the output of FTM4 channel 1 PWM/OCMP 1 FTM4CH1 pin is the output of FTM4 channel 1 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT1
22 FTM4OCH0SRC	FTM4 channel 0 output PWM/OCMP mode source select 0 FTM4CH0 pin is the output of FTM4 channel 0 PWM/OCMP 1 FTM4CH0 pin is the output of FTM4 channel 0 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT1
21 FTM3OCH5SRC	FTM3 channel 5 output PWM/OCMP mode source select 0 FTM3CH5 pin is the output of FTM3 channel 5 PWM/OCMP 1 FTM3CH5 pin is the output of FTM3 channel 5 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT1
20 FTM3OCH4SRC	FTM3 channel 4 output PWM/OCMP mode source select

Table continues on the next page...

SIM_SOPT9 field descriptions (continued)

Field	Description
	0 FTM3CH4 pin is the output of FTM3 channel 4 PWM/OCMP 1 FTM3CH4 pin is the output of FTM3 channel 4 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT1
19 FTM3OCH3SRC	FTM3 channel 3 output PWM/OCMP mode source select 0 FTM3CH3 pin is the output of FTM3 channel 3 PWM/OCMP 1 FTM3CH3 pin is the output of FTM3 channel 3 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT1
18 FTM3OCH2SRC	FTM3 channel 2 output PWM/OCMP mode source select 0 FTM3CH2 pin is the output of FTM3 channel 2 PWM/OCMP 1 FTM3CH2 pin is the output of FTM3 channel 2 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT1
17 FTM3OCH1SRC	FTM3 channel 1 output PWM/OCMP mode source select 0 FTM3CH1 pin is the output of FTM3 channel 1 PWM/OCMP 1 FTM3CH1 pin is the output of FTM3 channel 1 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT1
16 FTM3OCH0SRC	FTM3 channel 0 output PWM/OCMP mode source select 0 FTM3CH0 pin is the output of FTM3 channel 0 PWM/OCMP 1 FTM3CH0 pin is the output of FTM3 channel 0 PWM/OCMP modulating the carrier frequency, as per CARRIER_SELECT1
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 CARRIER_ SELECT1	Carrier frequency selection for FTM3/4 output channel 00 FTM1_CH1 output provides the carrier signal for Timer Modulation mode 01 LPTMR0 prescaler output provides the carrier signal for Timer Modulation mode 10 FTM5_CH1 output provides the carrier signal for Timer Modulation mode 11 Reserved
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.9 System Device Identification Register (SIM_SDID)

Address: 4004_7000h base + 1024h offset = 4004_8024h

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	FAMID				SUBFAMID				SERIERID				SRAMSIZE				REVID				DIEID					0			PINID			
W																																
Reset	0	0	0	1	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	0	0	0	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

SIM_SDID field descriptions

Field	Description
31–28 FAMID	V-Family ID Specifies the V-family of the device. 0001 MKV1xZx
27–24 SUBFAMID	V Sub-family ID Specifies the V sub-family of the device. 0000 MKV10xxxx 0001 MKV11xxxx
23–20 SERIERID	Series ID 0110 V-family - Motor control
19–16 SRAMSIZE	Specifies the size of the System SRAM. 0000 Reserved 0001 Reserved 0010 Reserved 0011 Reserved 0100 Reserved 0101 16 KB 0110 Reserved 0111 Reserved
15–12 REVID	Device revision number Specifies the silicon implementation number for the device.
11–7 DIEID	Device die number Specifies the silicon implementation number for the device. This field always reads as 00001.
6–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PINID	Pincount identification Specifies the pincount of the device. 0000 Reserved 0001 Reserved 0010 32-pin 0011 Reserved 0100 48-pin 0101 64-pin 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved

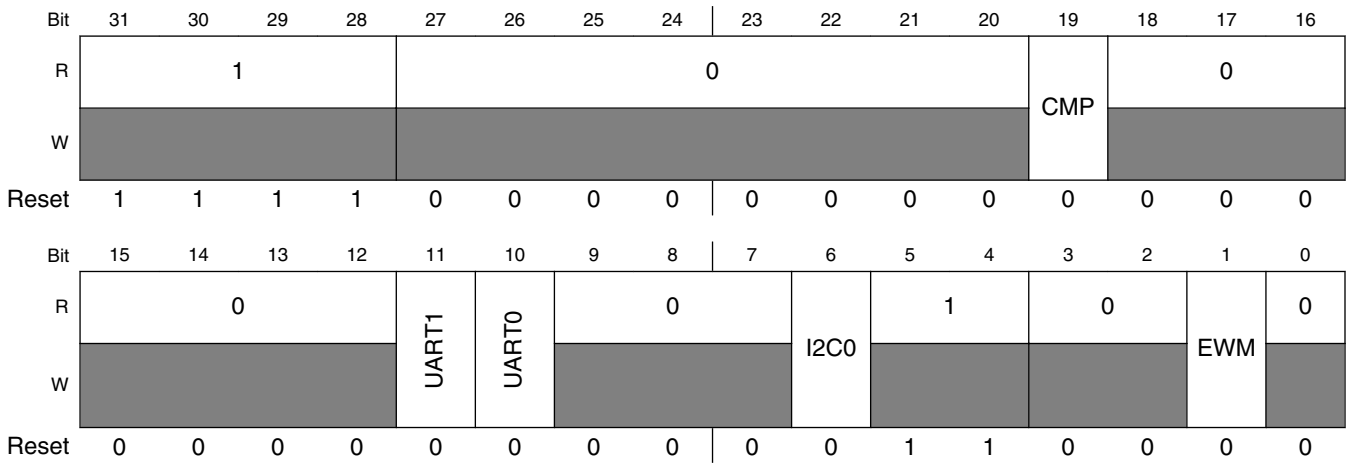
Table continues on the next page...

SIM_SDID field descriptions (continued)

Field	Description
	1101 Reserved 1110 Reserved 1111 Reserved

12.2.10 System Clock Gating Control Register 4 (SIM_SCGC4)

Address: 4004_7000h base + 1034h offset = 4004_8034h



SIM_SCGC4 field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
27–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 CMP	Comparator Clock Gate Control Controls the clock gate to both the comparator modules (CMP0 and CMP1). 0 Clock disabled 1 Clock enabled
18–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 UART1	UART1 Clock Gate Control This bit controls the clock gate to the UART1 module. 0 Clock disabled 1 Clock enabled
10 UART0	UART0 Clock Gate Control This bit controls the clock gate to the UART0 module.

Table continues on the next page...

SIM_SCGC4 field descriptions (continued)

Field	Description
	0 Clock disabled 1 Clock enabled
9–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 I2C0	I2C0 Clock Gate Control This bit controls the clock gate to the I^2C0 module. 0 Clock disabled 1 Clock enabled
5–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 EWM	EWM Clock Gate Control This bit controls the clock gate to the EWM module. 0 Clock disabled 1 Clock enabled
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.11 System Clock Gating Control Register 5 (SIM_SCGC5)

Address: 4004_7000h base + 1038h offset = 4004_8038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													1	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		PORTE	PORTD	PORTC	PORTB	PORTA	1	0	0						LPTMR
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

SIM_SCGC5 field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

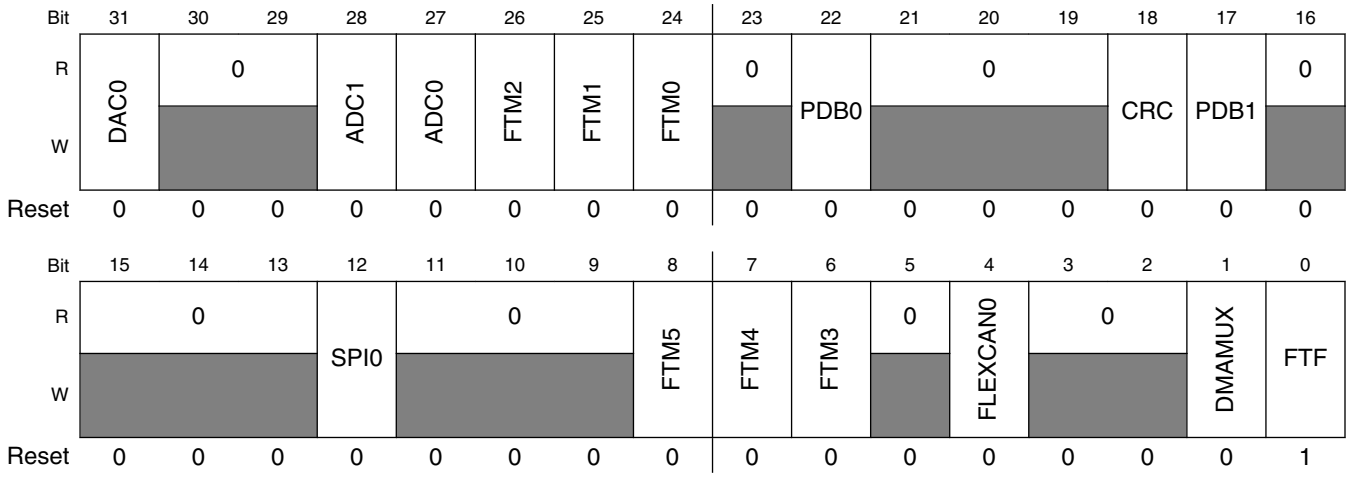
Table continues on the next page...

SIM_SCGC5 field descriptions (continued)

Field	Description
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 PORTE	Port E Clock Gate Control Controls the clock gate to the Port E module. 0 Clock disabled 1 Clock enabled
12 PORTD	Port D Clock Gate Control Controls the clock gate to the Port D module. 0 Clock disabled 1 Clock enabled
11 PORTC	Port C Clock Gate Control Controls the clock gate to the Port C module. 0 Clock disabled 1 Clock enabled
10 PORTB	Port B Clock Gate Control Controls the clock gate to the Port B module. 0 Clock disabled 1 Clock enabled
9 PORTA	Port A Clock Gate Control Controls the clock gate to the Port A module. 0 Clock disabled 1 Clock enabled
8–7 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.
5–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 LPTMR	Low Power Timer Clock Gate Control Controls the clock software access to the Low Power Timer module. 0 Access disabled 1 Access enabled

12.2.12 System Clock Gating Control Register 6 (SIM_SCGC6)

Address: 4004_7000h base + 103Ch offset = 4004_803Ch



SIM_SCGC6 field descriptions

Field	Description
31 DAC0	DAC0 Clock Gate Control Controls the clock gate to the DAC0 module. 0 Clock disabled 1 Clock enabled
30–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 ADC1	ADC1 Clock Gate Control Controls the clock gate to the ADC1 module. 0 Clock disabled 1 Clock enabled
27 ADC0	ADC0 Clock Gate Control Controls the clock gate to the ADC0 module. 0 Clock disabled 1 Clock enabled
26 FTM2	FTM2 Clock Gate Control Controls the clock gate to the FTM2 module. 0 Clock disabled 1 Clock enabled
25 FTM1	FTM1 Clock Gate Control Controls the clock gate to the FTM1 module.

Table continues on the next page...

SIM_SCGC6 field descriptions (continued)

Field	Description
	0 Clock disabled 1 Clock enabled
24 FTM0	FTM0 Clock Gate Control Controls the clock gate to the FTM0 module. 0 Clock disabled 1 Clock enabled
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 PDB0	PDB0 Clock Gate Control Controls the clock gate to the PDB module. 0 Clock disabled 1 Clock enabled
21–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 CRC	CRC Clock Gate Control Controls the clock gate to the CRC module. 0 Clock disabled 1 Clock enabled
17 PDB1	PDB1 Clock Gate Control 0 Clock disabled 1 Clock enabled
16–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 SPI0	SPI0 Clock Gate Control Controls the clock gate to the SPI0 module. 0 Clock disabled 1 Clock enabled
11–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 FTM5	FTM5 Clock Gate Control Controls the clock gate to the FTM5 module. 0 Clock disabled 1 Clock enabled
7 FTM4	FTM4 Clock Gate Control Controls the clock gate to the FTM4 module. 0 Clock disabled 1 Clock enabled

Table continues on the next page...

SIM_SCGC6 field descriptions (continued)

Field	Description
6 FTM3	FTM3 Clock Gate Control Controls the clock gate to the FTM3 module. 0 Clock disabled 1 Clock enabled
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 FLEXCAN0	FLEXCAN0 Clock Gate Control Controls the clock gate to the FLEXCAN0 module. 0 Clock disabled 1 Clock enabled
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DMAMUX	DMA Mux Clock Gate Control Controls the clock gate to the DMA Mux module. 0 Clock disabled 1 Clock enabled
0 FTF	Flash Memory Clock Gate Control Controls the clock gate to the flash memory. Flash reads are still supported while the flash memory is clock gated, but entry into low power modes is blocked. 0 Clock disabled 1 Clock enabled

12.2.13 System Clock Gating Control Register 7 (SIM_SCGC7)

Address: 4004_7000h base + 1040h offset = 4004_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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							DMA	0							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

SIM_SCGC7 field descriptions

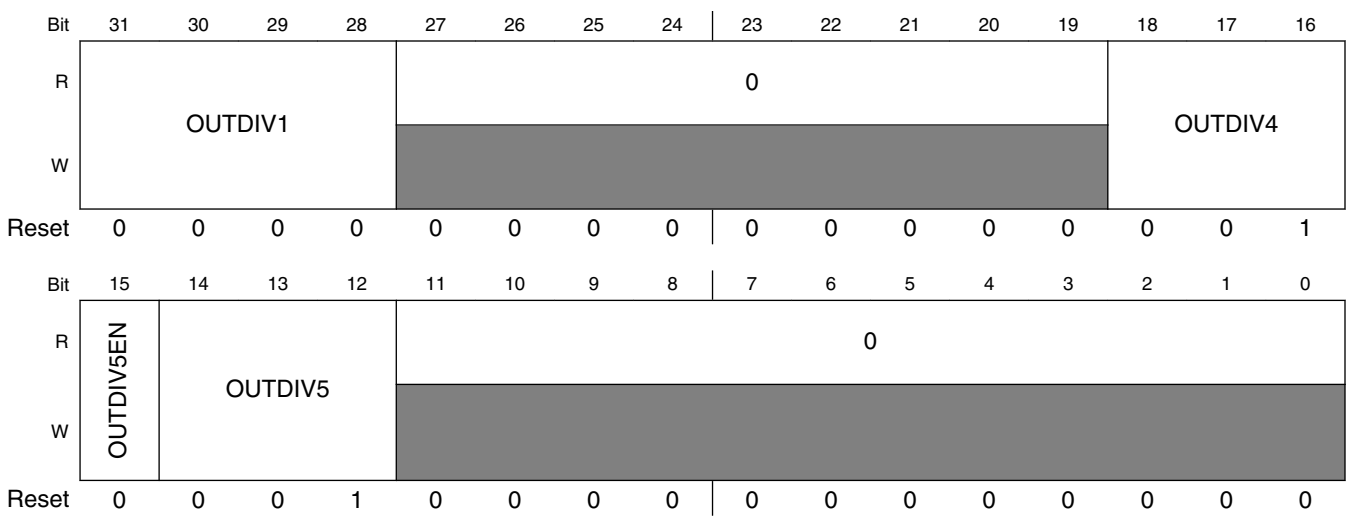
Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 DMA	DMA Clock Gate Control Controls the clock gate to the DMA module. 0 Clock disabled 1 Clock enabled
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.14 System Clock Divider Register 1 (SIM_CLKDIV1)

NOTE

The CLKDIV1 register cannot be written to when the device is in VLPR mode.

Address: 4004_7000h base + 1044h offset = 4004_8044h



SIM_CLKDIV1 field descriptions

Field	Description
31–28 OUTDIV1	Clock 1 Output Divider Value This field sets the divide value for the core/system clock. At the end of reset, it is loaded with 0000 (divide by one), 0001 (divide by two), 0011 (divide by four), or 0111 (divide by eight) depending on the setting of the two FTF_FOPT[LPBOOT] configuration bits. 0000 Divide-by-1. 0001 Divide-by-2.

Table continues on the next page...

SIM_CLKDIV1 field descriptions (continued)

Field	Description
	0010 Divide-by-3. 0011 Divide-by-4. 0100 Divide-by-5. 0101 Divide-by-6. 0110 Divide-by-7. 0111 Divide-by-8. 1000 Divide-by-9. 1001 Divide-by-10. 1010 Divide-by-11. 1011 Divide-by-12. 1100 Divide-by-13. 1101 Divide-by-14. 1110 Divide-by-15. 1111 Divide-by-16.
27–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–16 OUTDIV4	Clock 4 Output Divider Value This field sets the divide value for the bus and flash clock and is in addition to the System clock divide ratio. At the end of reset, it is loaded with 001 (divide by two). 000 Divide-by-1. 001 Divide-by-2. 010 Divide-by-3. 011 Divide-by-4. 100 Divide-by-5. 101 Divide-by-6. 110 Divide-by-7. 111 Divide-by-8.
15 OUTDIV5EN	OUTDIV5 Divider Control This bit controls the OUTDIV5 divider. 0 OUTDIV5 disabled 1 OUTDIV5 enabled
14–12 OUTDIV5	Clock 5 Output Divider Value Sets the divide value for the alternative ADC clock and is in addition to the System clock divide ratio. At the end of reset, it is loaded with 001 (divide by two). 000 Divide-by-1 001 Divide-by-2 010 Divide-by-3 011 Divide-by-4 100 Divide-by-5 101 Divide-by-6 110 Divide-by-7 111 Divide-by-8
Reserved	This field is reserved.

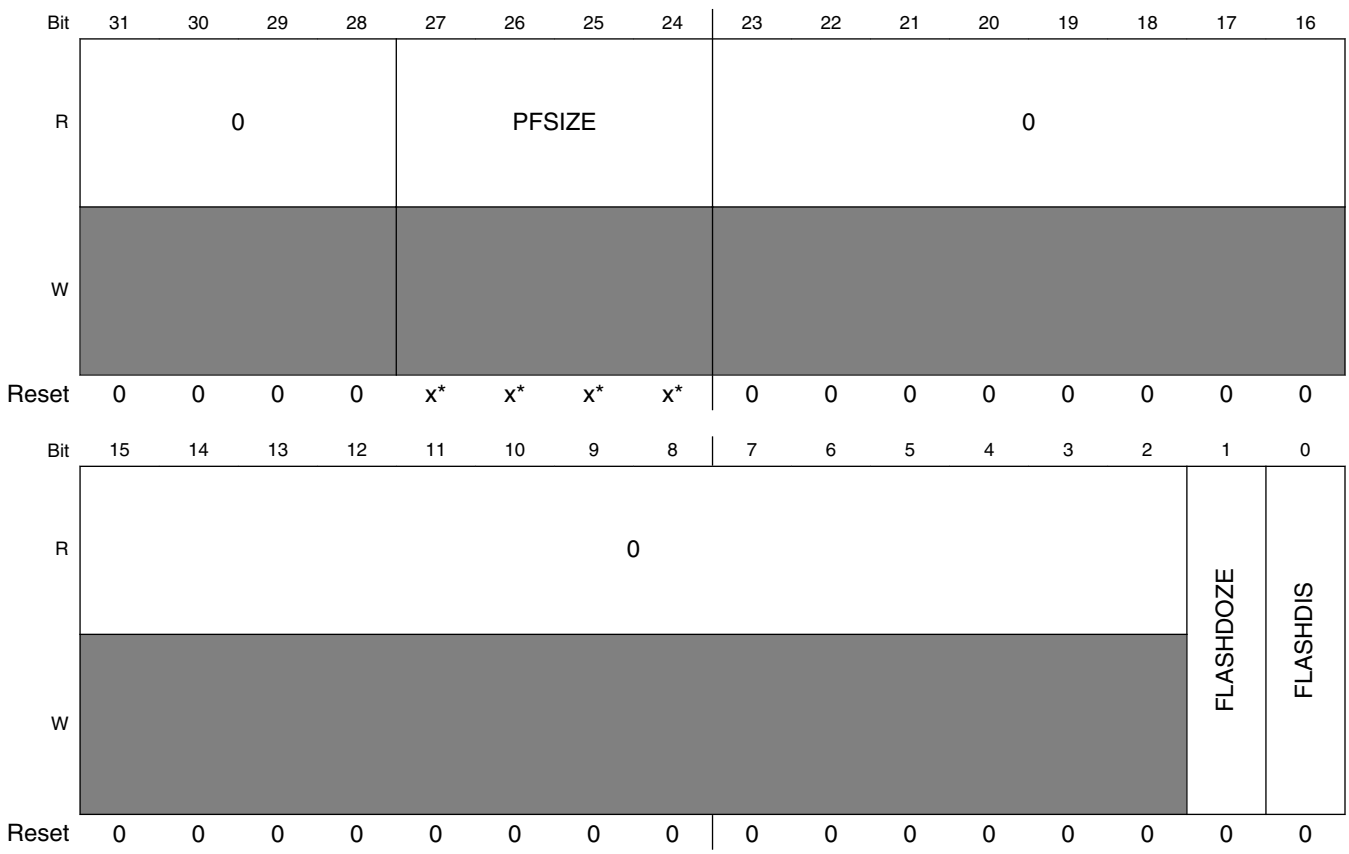
Table continues on the next page...

SIM_CLKDIV1 field descriptions (continued)

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

12.2.15 Flash Configuration Register 1 (SIM_FCFG1)

Address: 4004_7000h base + 104Ch offset = 4004_804Ch



* Notes:
- x = Undefined at reset.

SIM_FCFG1 field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–24 PFSIZE	Program Flash Size This field specifies the amount of program flash memory available on the device.

Table continues on the next page...

SIM_FCFG1 field descriptions (continued)

Field	Description
	0000 8 KB of program flash memory, 0.25 KB protection region 0001 16 KB of program flash memory, 0.5 KB protection region 0011 32 KB of program flash memory, 1 KB protection region 0101 64 KB of program flash memory, 2 KB protection region 0111 128 KB of program flash memory, 4 KB protection region 1001 256 KB of program flash memory, 4 KB protection region 1111 32 KB of program flash memory, 1 KB protection region
23–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FLASHDOZE	Flash Doze When set, Flash memory is disabled for the duration of Doze mode. An attempt by the DMA or other bus master to access the Flash when the Flash is disabled will result in a bus error. This field should be clear during VLP modes. The Flash will be automatically enabled again at the end of Doze mode so interrupt vectors do not need to be relocated out of Flash memory. The wakeup time from Doze mode is extended when this field is set. 0 Flash remains enabled during Doze mode 1 Flash is disabled for the duration of Doze mode
0 FLASHDIS	Flash Disable Flash accesses are disabled (and generate a bus error) and the Flash memory is placed in a low power state. This field should not be changed during VLP modes. Relocate the interrupt vectors out of Flash memory before disabling the Flash. 0 Flash is enabled 1 Flash is disabled

12.2.16 Flash Configuration Register 2 (SIM_FCFG2)

Address: 4004_7000h base + 1050h offset = 4004_8050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MAXADDR							1	0						
W																
Reset	0	x*	x*	x*	x*	x*	x*	x*	1	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															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- x = Undefined at reset.

SIM_FCFG2 field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–24 MAXADDR	Max address block This field concatenated with 13 leading zeros indicates the first invalid address of program flash. For example, if MAXADDR = 0x10, the first invalid address of flash block 0 is 0x0002_0000. This would be the MAXADDR value for a device with 128 KB program flash.
23 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.

12.2.17 Unique Identification Register Mid-High (SIM_UIDMH)

Address: 4004_7000h base + 1058h offset = 4004_8058h

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																UID															
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*

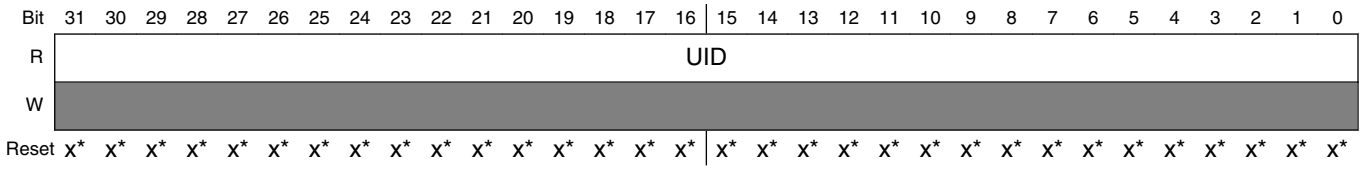
* Notes:
- x = Undefined at reset.
- x = Undefined at reset.

SIM_UIDMH field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
UID	Unique Identification Unique identification for the device.

12.2.18 Unique Identification Register Mid Low (SIM_UIDML)

Address: 4004_7000h base + 105Ch offset = 4004_805Ch



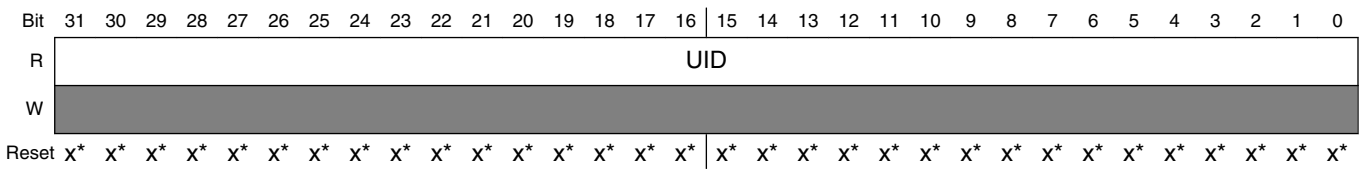
* Notes:
- x = Undefined at reset.

SIM_UIDML field descriptions

Field	Description
UID	Unique Identification Unique identification for the device.

12.2.19 Unique Identification Register Low (SIM_UIDL)

Address: 4004_7000h base + 1060h offset = 4004_8060h



* Notes:
- x = Undefined at reset.

SIM_UIDL field descriptions

Field	Description
UID	Unique Identification Unique identification for the device.

12.2.20 WDOG Control Register (SIM_WDOGC)

Address: 4004_7000h base + 1100h offset = 4004_8100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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														WDOGCLKS	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_WDOGC field descriptions

Field	Description
31–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 WDOGCLKS	WDOG Clock Select Selects the default clock source of the WDOG watchdog. On reset it will be the 1kHZ LPO. The software can change this to the MCGIRCLK, which can be 32KHz or 4Mhz, depending on the MCG configuration. 0 Internal 1 kHz clock is source to WDOG 1 MCGIRCLK is source to WDOG
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.3 Functional description

For more information about the functions of SIM, see the Introduction section.

Chapter 13
Kinetis Flashloader

13.1 Introduction

The Kinetis devices *that do not have an on-chip ROM* are shipped with the pre-programmed Kinetis Flashloader in the on-chip flash memory, for one-time, in-system factory programming. The Kinetis Flashloader's main task is to load a customer firmware image into the flash memory. The image on the flash has 2 programs: flashloader_loader and flashloader. After a device reset, the flashloader_loader program starts its execution first. The flashloader_loader program copies the contents of flashloader image from the flash to the on-chip RAM; the device then switches execution to the flashloader program to execute from RAM.

For this device, the Kinetis Flashloader can interface with UART, CAN, I2C, and SPI peripherals in slave mode and respond to the commands sent by a master (or host) communicating on one of those ports. The host/master can be a firmware-download application running on a PC or an embedded host communicating with the Kinetis Flashloader. Regardless of the host/master (PC or embedded host), the Kinetis Flashloader always uses a command protocol to communicate with that host/master. Commands are provided to write to memory (flash or RAM), erase flash, and get/set flashloader options and property values. The host application can query the set of available commands.

This chapter describes Kinetis Flashloader features, functionality, command structure and which peripherals are supported.

Features supported by the Kinetis Flashloader :

- Supports UART, CAN, I2C, and SPI peripheral interfaces
- Automatic detection of the active peripheral
- UART and CAN peripherals with autobaud
- Common packet-based protocol for all peripherals
- Packet error detection and retransmission

- Fully supports flash security, including ability to mass erase or unlock security via the backdoor key
- Protection of RAM used by the flashloader while it is running
- Provides command to read properties of the device, such as flash and RAM size

Table 13-1. Commands supported by the Kinetis Flashloader

Command	Description	When flash security is enabled, then this command is
Execute	Run user application code that never returns control to the flashloader	Not supported
FillMemory	Fill a range of bytes in flash with a word pattern	Not supported
FlashEraseAll	Erase the entire flash array	Not supported
FlashEraseRegion	Erase a range of sectors in flash	Not supported
WriteMemory	Write data to memory	Not supported
ReadMemory	Read data from memory	Not supported
FlashSecurityDisable	Attempt to unlock flash security using the backdoor key	Supported
GetProperty	Get the current value of a property	Supported
Reset	Reset the chip	Supported
SetProperty	Attempt to modify a writable property	Supported

13.2 Functional Description

The following sub-sections describe the Kinetis Flashloader functionality.

13.2.1 Memory Maps

While executing, the Kinetis Flashloader uses RAM memory.

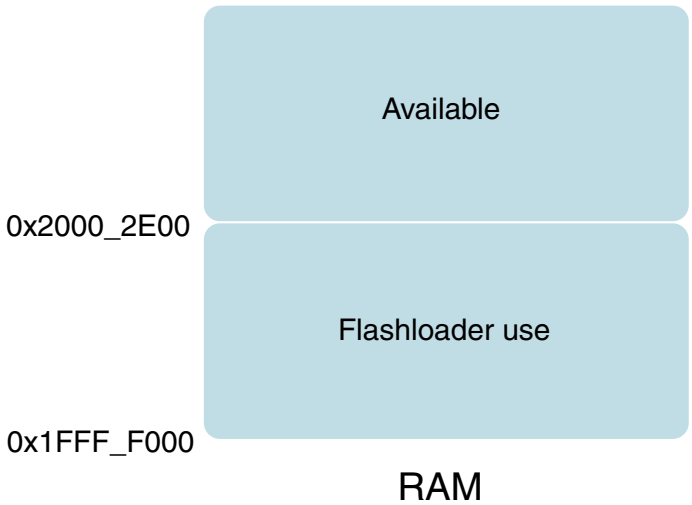


Figure 13-1. Kinetis Flashloader RAM Memory Map

NOTE

The Kinetis Flashloader requires a minimum memory space of 16 KB of RAM. For Kinetis devices with less than this amount of on-chip RAM, the Kinetis Flashloader is not available.

13.2.2 Start-up Process

As the Kinetis Flashloader begins executing, flashloader operations begin:

1. The flashloader's temporary working area in RAM is initialized.
2. All supported peripherals are initialized.
3. The flashloader waits for communication to begin on a peripheral.
   - There is no timeout for the active peripheral detection process.
   - If communication is detected, then all inactive peripherals are shut down, and the command phase is entered.

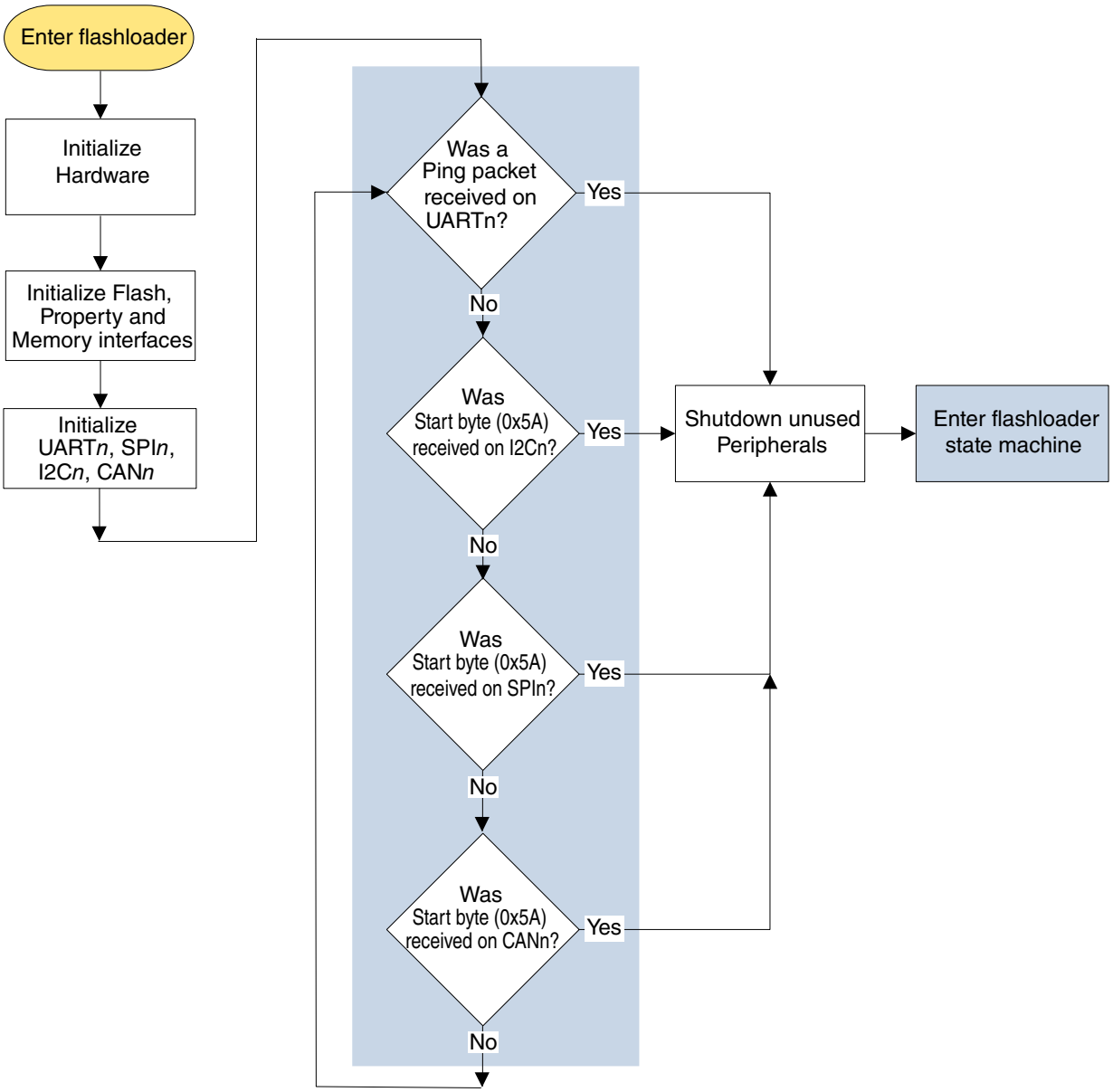


Figure 13-2. Kinetis Flashloader Start-up Flowchart

13.2.3 Clock Configuration

13.2.4 Flashloader Protocol

This section explains the general protocol for the packet transfers between the host and the Kinetis Flashloader. The description includes the transfer of packets for different transactions, such as commands with no data phase and commands with incoming or outgoing data phase. The next section describes various packet types used in a transaction.

Each command sent from the host is replied to with a response command.

Commands may include an optional data phase:

- If the data phase is **incoming** (from host to flashloader), then the data phase is part of the **original command**.
- If the data phase is **outgoing** (from flashloader to host), then the data phase is part of the **response command**.

NOTE

In all protocols (described in the next subsections), the Ack sent in response to a Command or Data packet can arrive at any time *before, during, or after* the Command/Data packet has processed.

13.2.4.1 Command with no data phase

The protocol for a command with no data phase contains:

- Command packet (from host)
- Generic response command packet (to host)

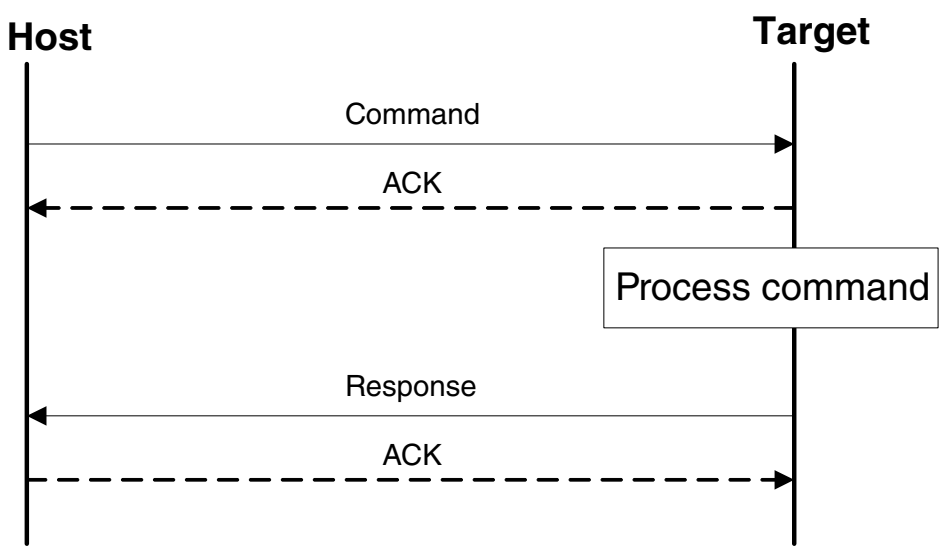


Figure 13-3. Command with No Data Phase

13.2.4.2 Command with incoming data phase

The protocol for a command with an incoming data phase contains:

- Command packet (from host)
- Generic response command packet (to host)
- Incoming data packets (from host)
- Generic response command packet (to host)

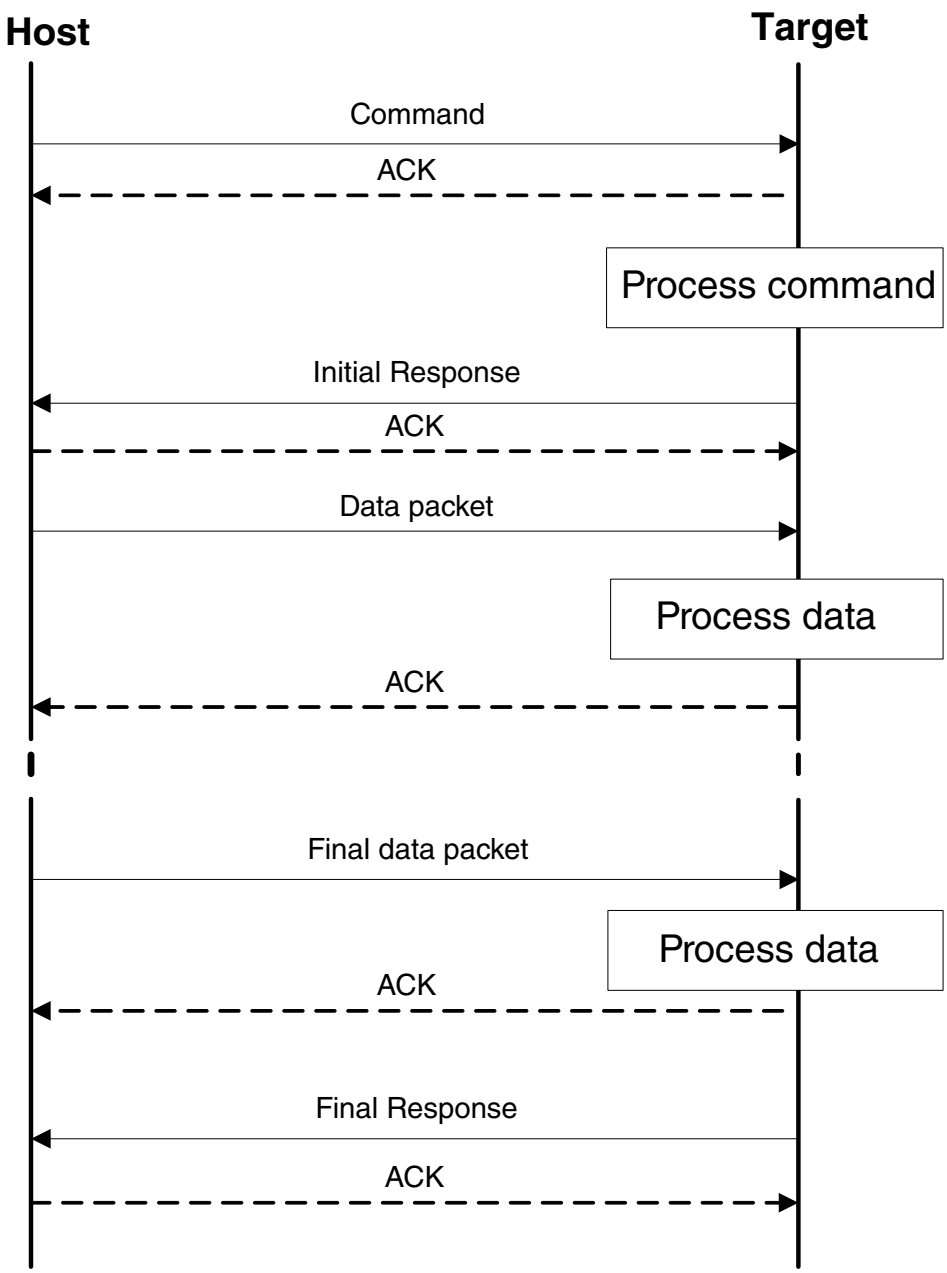


Figure 13-4. Command with incoming data phase

NOTE

- The host may not send any further packets while it (the host) is waiting for the response to a command.
- If the Generic Response packet prior to the start of the data phase does not have a status of kStatus_Success, then the data phase is aborted.
- Data phases may be aborted by the receiving side by sending the final Generic Response early with a status of kStatus_AbortDataPhase. The host may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet *sent after the data phase* includes the status for the entire operation.

13.2.4.3 Command with outgoing data phase

The protocol for a command with an outgoing data phase contains:

- Command packet (from host)
- ReadMemory Response command packet (to host) (kCommandFlag_HasDataPhase set)
- Outgoing data packets (to host)
- Generic response command packet (to host)

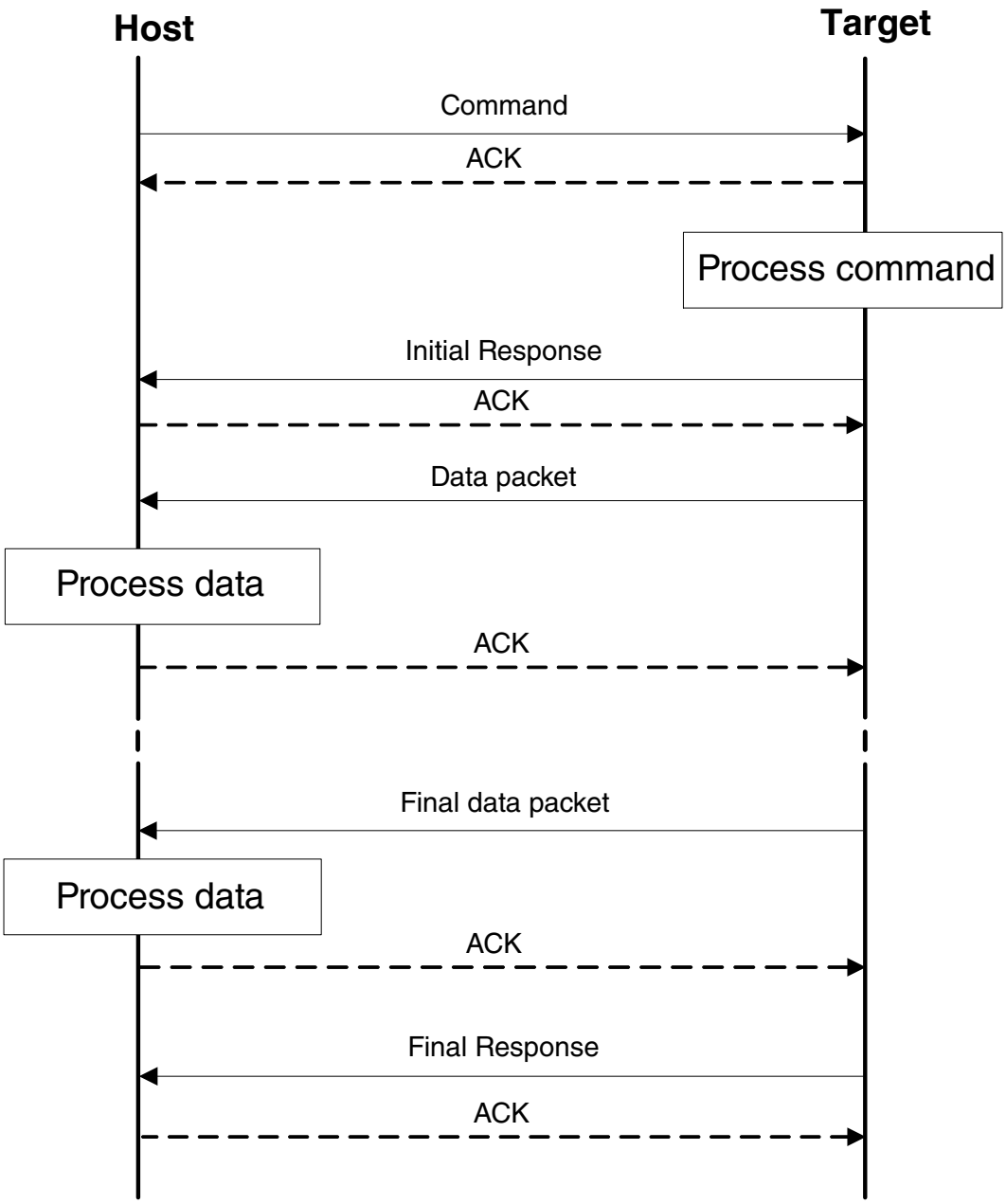


Figure 13-5. Command with outgoing data phase

NOTE

- For the outgoing data phase sequence above, the data phase is really considered part of the response command.
- The host may not send any further packets while it (the host) is waiting for the response to a command.
- If the ReadMemory Response command packet prior to the start of the data phase does not contain the kCommandFlag_HasDataPhase flag, then the data phase is aborted.

- Data phases may be aborted by the host sending the final Generic Response early with a status of kStatus_AbortDataPhase. The sending side may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet *sent after the data phase* includes the status for the entire operation.

13.2.5 Flashloader Packet Types

The Kinetis Flashloader device works in slave mode. All data communication is initiated by a host, which is either a PC or an embedded host . The Kinetis Flashloader device is the target, which receives a command or data packet. All data communication between host and target is packetized.

NOTE

The term "target" refers to the "Kinetis Flashloader device."

There are 6 types of packets used in the device:

- Ping packet
- Ping Response packet
- Framing packet
- Command packet
- Data packet
- Response packet

All fields in the packets are in little-endian byte order.

13.2.5.1 Ping packet

The Ping packet is the first packet sent from a host to the target (Kinetis Flashloader), to establish a connection on a selected peripheral. For a UART peripheral, the Ping packet is used to determine the baudrate. A Ping packet must be sent before any other communications. In response to a Ping packet, the target sends a Ping Response packet.

Table 13-2. Ping Packet Format

Byte #	Value	Name
0	0x5A	start byte
1	0xA6	ping

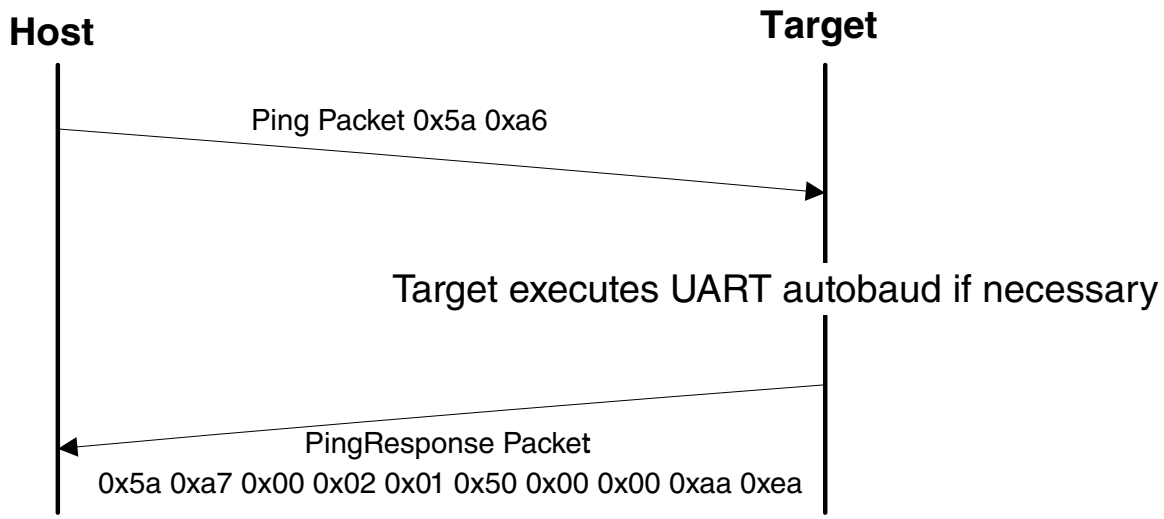


Figure 13-6. Ping Packet Protocol Sequence

13.2.5.2 Ping Response Packet

The target (Kinetis Flashloader) sends a Ping Response packet back to the host after receiving a Ping packet. If communication is over a UART peripheral, the target uses the incoming Ping packet to determine the baud rate before replying with the Ping Response packet. Once the Ping Response packet is received by the host, the connection is established, and the host starts sending commands to the target (Kinetis Flashloader).

Table 13-3. Ping Response Packet Format

Byte #	Value	Parameter
0	0x5A	start byte
1	0xA7	Ping response code
2		Protocol bugfix
3		Protocol minor
4		Protocol major
5		Protocol name = 'P' (0x50)
6		Options low
7		Options high
8		CRC16 low
9		CRC16 high

13.2.5.3 Framing Packet

The framing packet is used for flow control and error detection, and it (the framing packet) wraps command and data packets as well.

The framing packet described in this section is used for serial peripherals including UART, I2C, SPI, and CAN.

Table 13-4. Framing Packet Format

Byte #	Value	Parameter	
0	0x5A	start byte	
1		packetType	
2		length_low	Length is a 16-bit field that specifies the entire command or data packet size in bytes.
3		length_high	
4		crc16_low	This is a 16-bit field. The CRC16 value covers entire framing packet, including the start byte and command or data packets, but does not include the CRC bytes. See the CRC16 algorithm after this table.
5		crc16_high	
6 . . .n		Command or Data packet payload	

A special framing packet that contains only a start byte and a packet type is used for synchronization between the host and target.

Table 13-5. Special Framing Packet Format

Byte #	Value	Parameter
0	0x5A	start byte
1	0xA*n*	packetType

The Packet Type field specifies the type of the packet from one of the defined types (below):

Table 13-6. packetType Field

packetType	Name	Description
0xA1	kFramingPacketType_Ack	The previous packet was received successfully; the sending of more packets is allowed.
0xA2	kFramingPacketType_Nak	The previous packet was corrupted and must be re-sent.
0xA3	kFramingPacketType_AckAbort	Data phase is being aborted.
0xA4	kFramingPacketType_Command	The framing packet contains a command packet payload.
0xA5	kFramingPacketType_Data	The framing packet contains a data packet payload.
0xA6	kFramingPacketType_Ping	Sent to verify the other side is alive. Also used for UART autobaud.

Table continues on the next page...

Table 13-6. packetType Field (continued)

packetType	Name	Description
0xA7	kFramingPacketType_PingResponse	A response to Ping; contains the framing protocol version number and options.

13.2.5.4 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

Table 13-7. Command Packet Format

Command Packet Format (32 bytes)										
Command Header (4 bytes)				28 bytes for Parameters (Max 7 parameters)						
Tag	Flags	Rsvd	Param Count	Param1 (32-bit)	Param2 (32-bit)	Param3 (32-bit)	Param4 (32-bit)	Param5 (32-bit)	Param6 (32-bit)	Param7 (32-bit)
byte 0	byte 1	byte 2	byte 3							

Table 13-8. Command Header Format

Byte #	Command Header Field	
0	Command or Response tag	The command header is 4 bytes long, with these fields.
1	Flags	
2	Reserved. Should be 0x00.	
3	ParameterCount	

The header is followed by 32-bit parameters up to the value of the ParameterCount field specified in the header. Because a command packet is 32 bytes long, only 7 parameters can fit into the command packet.

Command packets are also used by the target to send responses back to the host. As mentioned earlier, command packets and data packets are embedded into framing packets for all of the transfers.

Table 13-9. Commands that are supported

Command	Name
0x01	FlashEraseAll
0x02	FlashEraseRegion
0x03	ReadMemory
0x04	WriteMemory
0x05	FillMemory
0x06	FlashSecurityDisable

Table continues on the next page...

Table 13-9. Commands that are supported (continued)

Command	Name
0x07	GetProperty
0x08	Reserved
0x09	Execute
0x0A	Reserved
0x0B	Reset
0x0C	SetProperty
0x0D	Reserved
0x0E	Reserved
0x0F	Reserved
0x10	Reserved
0x11	Reserved
0x12	Reserved

Table 13-10. Responses that are supported

Response	Name
0xA0	GenericResponse
0xA3	ReadMemoryResponse (used for sending responses to ReadMemory command only)
0xA7	GetPropertyResponse (used for sending responses to GetProperty command only)

Flags: Each command packet contains a Flag byte. Only bit 0 of the flag byte is used. If bit 0 of the flag byte is set to 1, then data packets will follow in the command sequence. The number of bytes that will be transferred in the data phase is determined by a command-specific parameter in the parameters array.

ParameterCount: The number of parameters included in the command packet.

Parameters: The parameters are word-length (32 bits). With the default maximum packet size of 32 bytes, a command packet can contain up to 7 parameters.

13.2.5.5 Data packet

The data packet carries just the data, either host sending data to target, or target sending data to host. The data transfer direction is determined by the last command sent from the host. The data packet is also wrapped within a framing packet, to ensure the correct packet data is received.

The contents of a data packet are simply the data itself. There are no other fields, so that the most data per packet can be transferred. Framing packets are responsible for ensuring that the correct packet data is received.

13.2.5.6 Response packet

The responses are carried using the same command packet format wrapped with framing packet data. Types of responses include:

- GenericResponse
- GetPropertyResponse
- ReadMemoryResponse

GenericResponse: After the Kinetis Flashloader has processed a command, the flashloader will send a generic response with status and command tag information to the host. The generic response is the last packet in the command protocol sequence. The generic response packet contains the framing packet data and the command packet data (with generic response tag = 0xA0) and a list of parameters (defined in the next section). The parameter count field in the header is always set to 2, for status code and command tag parameters.

Table 13-11. GenericResponse Parameters

Byte #	Parameter	Descripton
0 - 3	Status code	The Status codes are errors encountered during the execution of a command by the target (Kinetis Flashloader). If a command succeeds, then a kStatus_Success code is returned. Table 13-37, Kinetis Flashloader Status Error Codes, lists the status codes returned to the host by the Kinetis Flashloader.
4 - 7	Command tag	The Command tag parameter identifies the response to the command sent by the host.

GetPropertyResponse: The GetPropertyResponse packet is sent by the target in response to the host query that uses the GetProperty command. The GetPropertyResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a GetPropertyResponse tag value (0xA7).

The parameter count field in the header is set to greater than 1, to always include the status code and one or many property values.

Table 13-12. GetPropertyResponse Parameters

Byte #	Value	Parameter
0 - 3		Status code
4 - 7		Property value
. . .		. . .
		Can be up to maximum 6 property values, limited to the size of the 32-bit command packet and property type.

ReadMemoryResponse: The ReadMemoryResponse packet is sent by the target in response to the host sending a ReadMemory command. The ReadMemoryResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a ReadMemoryResponse tag value (0xA3), the flags field set to kCommandFlag_HasDataPhase (1).

The parameter count set to 2 for the status code and the data byte count parameters shown below.

Table 13-13. ReadMemoryResponse Parameters

Byte #	Parameter	Descripton
0 - 3	Status code	The status of the associated Read Memory command.
4 - 7	Data byte count	The number of bytes sent in the data phase.

13.2.6 Flashloader Command API

All Kinetis Flashloader command APIs follow the command packet format that is wrapped by the framing packet, as explained in previous sections.

- For a list of commands supported by the Flashloader, see Table 13-1, Commands supported.
- For a list of status codes returned by the Kinetis Flashloader, see Table 13-37, Kinetis Flashloader Status Error Codes.

NOTE

All the examples in this section depict byte traffic on serial peripherals that use framing packets.

13.2.6.1 GetProperty command

The GetProperty command is used to query the flashloader about various properties and settings. Each supported property has a unique 32-bit tag associated with it. The tag occupies the first parameter of the command packet. The target returns a GetPropertyResponse packet with the property values for the property identified with the tag in the GetProperty command.

Properties are the defined units of data that can be accessed with the GetProperty or SetProperty commands. Properties may be read-only or read-write. All read-write properties are 32-bit integers, so they can easily be carried in a command parameter.

For a list of properties and their associated 32-bit property tags supported by the Kinetis Flashloader, see Table 13-33.

The 32-bit property tag is the only parameter required for GetProperty command.

Table 13-14. Parameters for GetProperty Command

Byte #	Command
0 - 3	Property tag

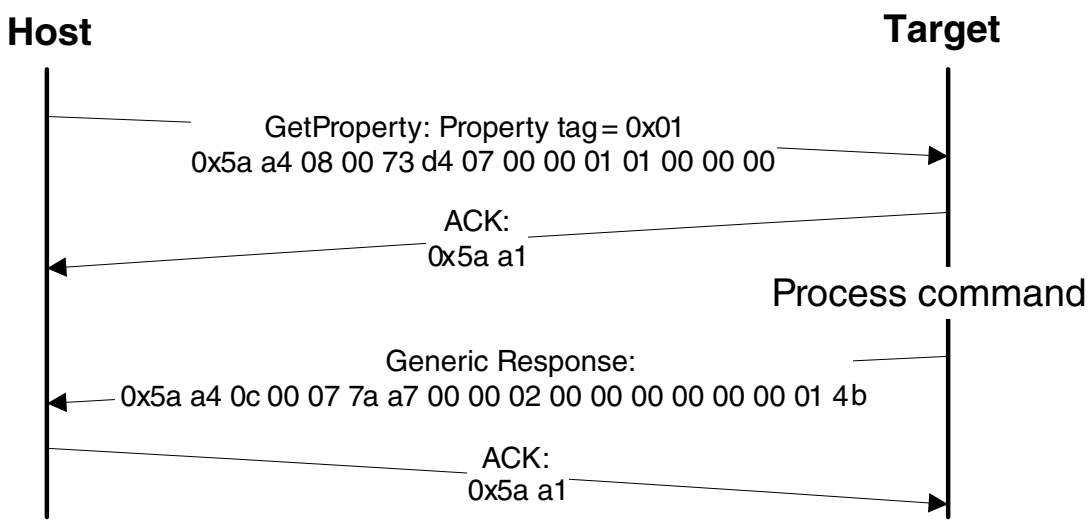


Figure 13-7. Protocol Sequence for GetProperty Command

Table 13-15. GetProperty Command Packet Format (Example)

GetProperty	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x08 0x00

Table continues on the next page...

Table 13-15. GetProperty Command Packet Format (Example) (continued)

GetProperty	Parameter	Value
	crc16	0x73 0xD4
Command packet	commandTag	0x07 – GetProperty
	flags	0x00
	reserved	0x00
	parameterCount	0x01
	propertyTag	0x00000001 - CurrentVersion

The GetProperty command has no data phase.

Response: In response to a GetProperty command, the target will send a GetPropertyResponse packet with the response tag set to 0xA7. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property value(s). The next table shows an example of a GetPropertyResponse packet.

Table 13-16. GetProperty Response Packet Format (Example)

GetPropertyResponse	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0c 0x00 (12 bytes)
	crc16	0x07 0x7a
Command packet	responseTag	0xA7
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	status	0x00000000
	propertyValue	0x0000014b - CurrentVersion

13.2.6.2 SetProperty command

The SetProperty command is used to change or alter the values of the properties or options in the Kinetis Flashloader. However, the SetProperty command can only change the value of properties that are writable—see Table 13-33, Properties used by Get/SetProperty Commands. If you try to set a value for a read-only property, then the Kinetis Flashloader will return an error.

The property tag and the new value to set are the 2 parameters required for the SetProperty command.

Table 13-17. Parameters for SetProperty Command

Byte #	Command
0 - 3	Property tag
4 - 7	Property value

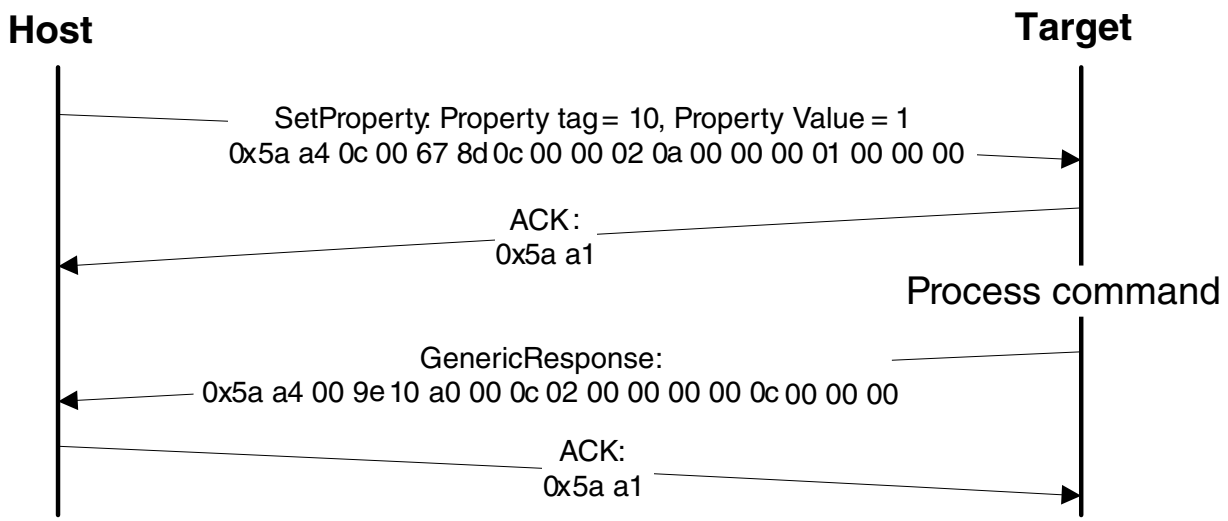


Figure 13-8. Protocol Sequence for SetProperty Command

Table 13-18. SetProperty Command Packet Format (Example)

SetProperty	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x67 0x8D
Command packet	commandTag	0x0C – SetProperty with property tag 10
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	propertyTag	0x0000000A - VerifyWrites
	propertyValue	0x00000001

The SetProperty command has no data phase.

Response: The target (Kinetis Flashloader) will return a GenericResponse packet with one of following status codes:

Table 13-19. SetProperty Response Status Codes

Status Code
kStatus_Success
kStatus_ReadOnly
kStatus_UnknownProperty
kStatus_InvalidArgument

13.2.6.3 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory. If any flash regions are protected, then the FlashEraseAll command will fail and return an error status code. Executing the FlashEraseAll command will release flash security if it (flash security) was enabled, by setting the FTFA_FSEC register. However, the FSEC field of the flash configuration field is erased, so unless it is reprogrammed, the flash security will be re-enabled after the next system reset. The Command tag for FlashEraseAll command is 0x01 set in the commandTag field of the command packet.

The FlashEraseAll command requires no parameters.

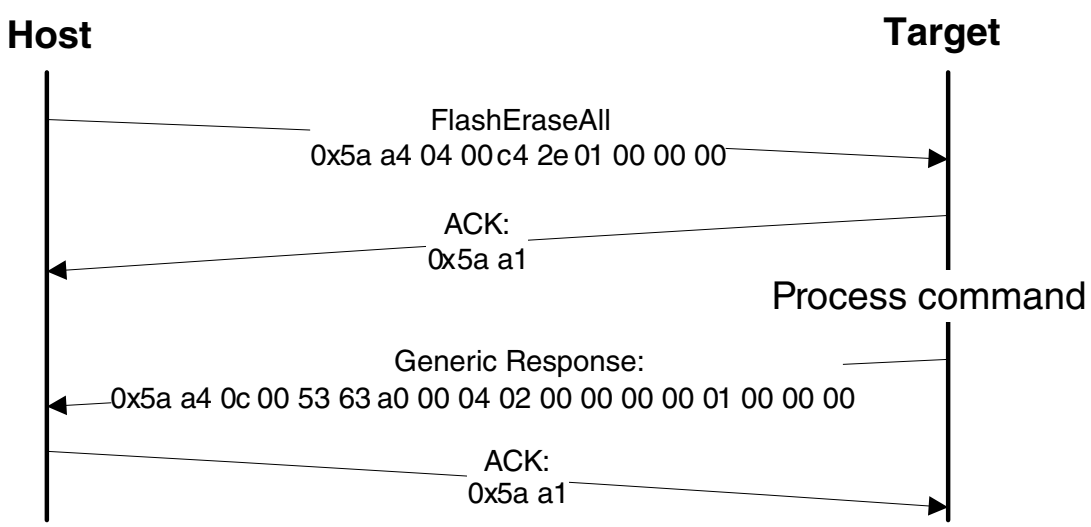


Figure 13-9. Protocol Sequence for FlashEraseAll Command

Table 13-20. FlashEraseAll Command Packet Format (Example)

FlashEraseAll	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x04 0x00
	crc16	0xC4 0x2E
Command packet	commandTag	0x01 - FlashEraseAll
	flags	0x00
	reserved	0x00
	parameterCount	0x00
	MemoryID	• If MemoryID = 0x00h, then internal flash. • If MemoryID = 0x01h, then QSPI0 memory.

The FlashEraseAll command has no data phase.

Response: The target (Kinetis Flashloader) will return a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command, or set to an appropriate error status code.

13.2.6.4 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory or a specified range of flash within the connected SPI flash devices.

The start address and number of bytes are the 2 parameters required for the FlashEraseRegion command. The start and byte count parameters must be , or the FlashEraseRegion command will fail and return kStatus_FlashAlignmentError (0x101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command will fail and return kStatus_FlashAddressError (0x102). If any part of the region specified is protected, the FlashEraseRegion command will fail and return kStatus_MemoryRangeInvalid (0x10200).

Table 13-21. Parameters for FlashEraseRegion Command

Byte #	Parameter
0 - 3	Start address
4 - 7	Byte count

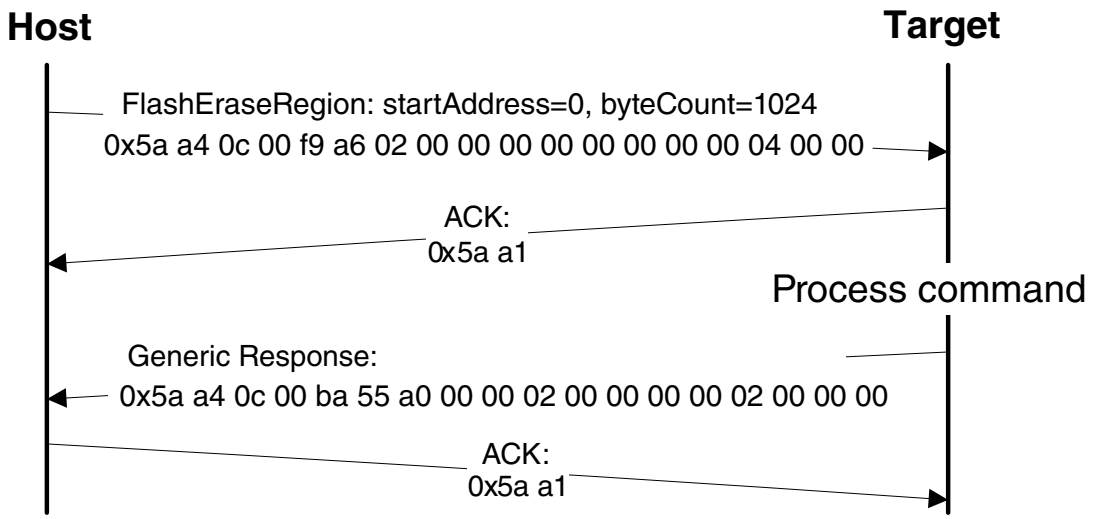


Figure 13-10. Protocol Sequence for FlashEraseRegion Command

Table 13-22. FlashEraseRegion Command Packet Format (Example)

FlashEraseRegion	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0xF9 0x A6
Command packet	commandTag	0x02, kCommandTag_FlashEraseRegion
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	startAddress	0x00 0x00 0x00 0x00 (0x0000_0000)
	byte count	0x00 0x04 0x00 0x00 (0x400)

The FlashEraseRegion command has no data phase.

Response: The target (Kinetis Flashloader) will return a GenericResponse packet with one of following error status codes.

Table 13-23. FlashEraseRegion Response Status Codes

Status Code
kStatus_Success (0x0)
kStatus_MemoryRangeInvalid (0x10200)
kStatus_FlashAlignmentError (0x101)
kStatus_FlashAddressError (0x102)
kStatus_FlashAccessError (0x103)
kStatus_FlashProtectionViolation (0x104)
kStatus_FlashCommandFailure (0x105)

13.2.6.5 FillMemory command

The FillMemory command fills a range of bytes in memory with a data pattern. It follows the same rules as the WriteMemory command. The difference between FillMemory and WriteMemory is that a data pattern is included in FillMemory command parameter, and there is no data phase for the FillMemory command, while WriteMemory does have a data phase.

Table 13-24. Parameters for FillMemory Command

Byte #	Command
0 - 3	Start address of memory to fill
4 - 7	Number of bytes to write with the pattern • The start address should be 32-bit aligned. • The number of bytes must be evenly divisible by 4.
8 - 11	32-bit pattern

- To fill with a byte pattern (8-bit), the byte must be replicated 4 times in the 32-bit pattern.
- To fill with a short pattern (16-bit), the short value must be replicated 2 times in the 32-bit pattern.

For example, to fill a byte value with 0xFE, the word pattern would be 0xFEFEFEFE; to fill a short value 0x5AFE, the word pattern would be 0x5AFE5AFE.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash requires the start address to be 32-bit or 64-bit aligned.
- If the VerifyWrites property is set to true, then writes to flash will also perform a flash verify program operation.

When writing to RAM, the start address need not be aligned, and the data will not be padded.

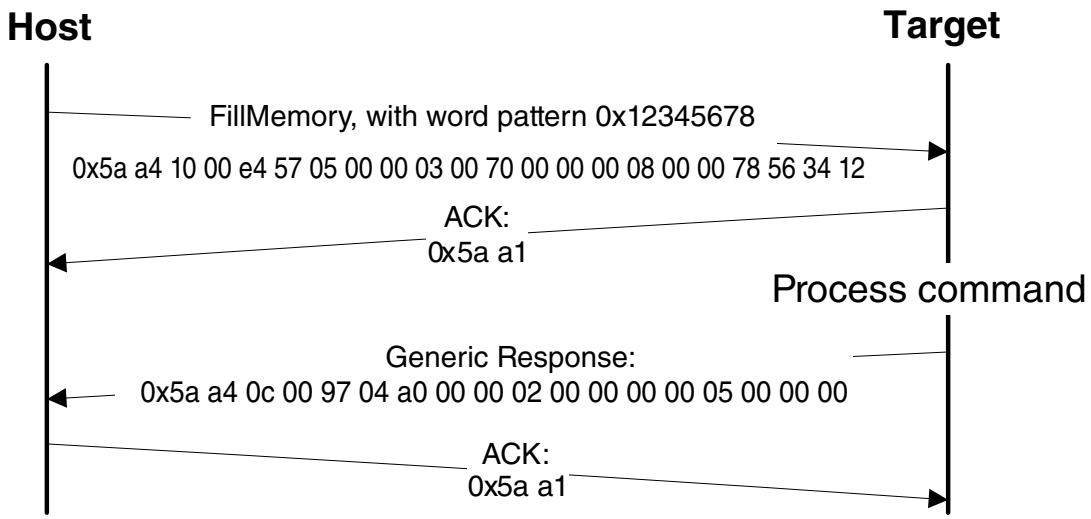


Figure 13-11. Protocol Sequence for FillMemory Command

Table 13-25. FillMemory Command Packet Format (Example)

FillMemory	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x10 0x00
	crc16	0xE4 0x57
Command packet	commandTag	0x05 – FillMemory
	flags	0x00
	Reserved	0x00
	parameterCount	0x03
	startAddress	0x00007000
	byteCount	0x00000800
	patternWord	0x12345678

The FillMemory command has no data phase.

Response: upon successful execution of the command, the target (Kinetis Flashloader) will return a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

13.2.6.6 WriteMemory command

The WriteMemory command writes data provided in the data phase to a specified range of bytes in memory (flash or RAM). However, if flash protection is enabled, then writes to protected sectors will fail.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash requires the start address to be .
- If the VerifyWrites property is set to true, then writes to flash will also perform a flash verify program operation.

When writing to RAM, the start address need not be aligned, and the data will not be padded.

The start address and number of bytes are the 2 parameters required for WriteMemory command.

Table 13-26. Parameters for WriteMemory Command

Byte #	Command
0 - 3	Start address
4 - 7	Byte count

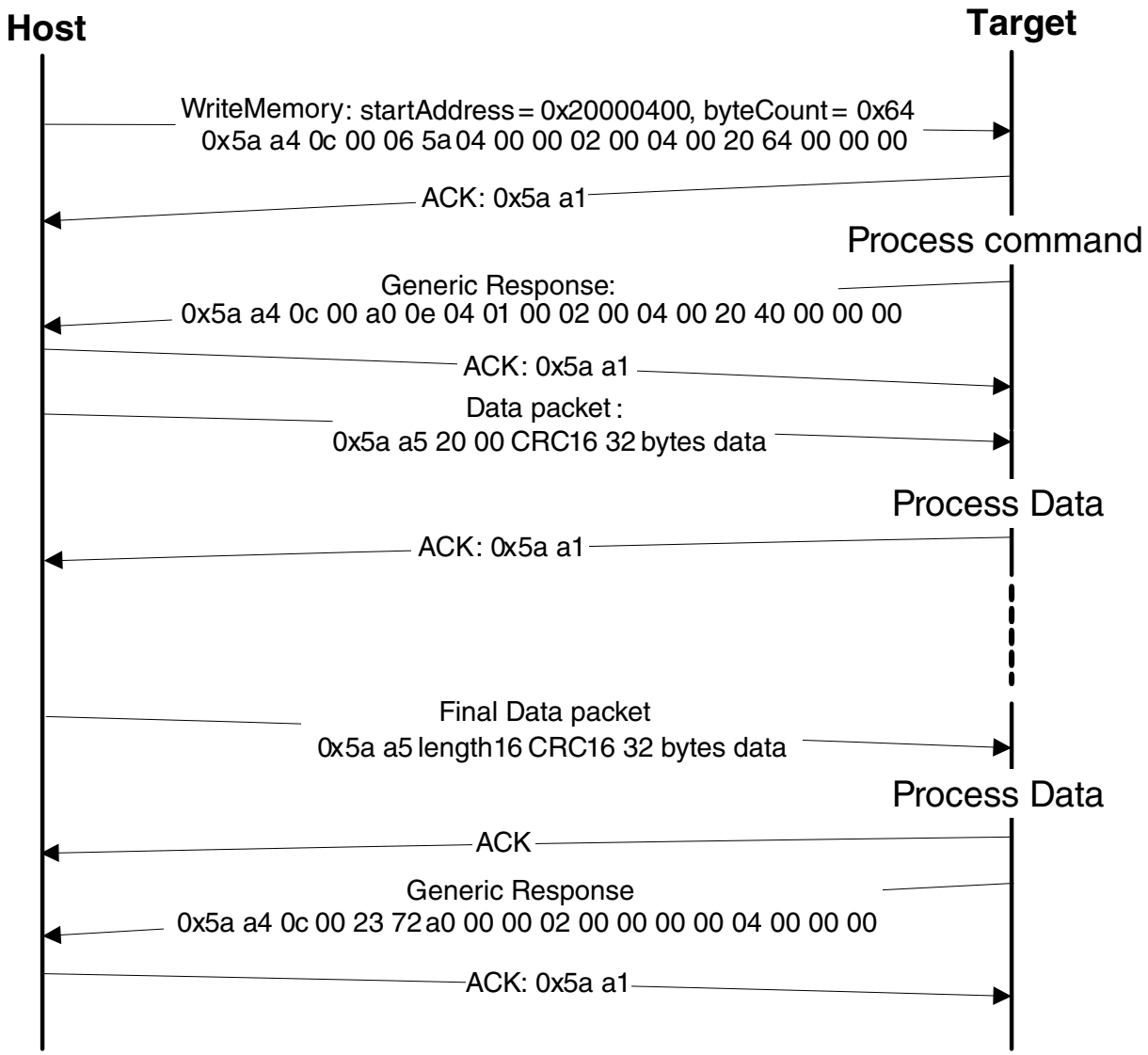


Figure 13-12. Protocol Sequence for WriteMemory Command

Table 13-27. WriteMemory Command Packet Format (Example)

WriteMemory	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x06 0x5A
Command packet	commandTag	0x04 - writeMemory
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	startAddress	0x20000400
	byteCount	0x00000064

Data Phase: The WriteMemory command has a data phase; the host will send data packets until the number of bytes of data specified in the byteCount parameter of the WriteMemory command are received by the target.

Response: The target (Kinetis Flashloader) will return a GenericResponse packet with a status code set to kStatus_Success upon successful execution of the command, or to an appropriate error status code.

13.2.6.7 Read memory command

The ReadMemory command returns the contents of memory at the given address, for a specified number of bytes. This command can read any region of memory accessible by the CPU and not protected by security.

The start address and number of bytes are the 2 parameters required for ReadMemory command.

Table 13-28. Parameters for read memory command

Byte	Parameter	Description
0-3	Start address	Start address of memory to read from
4-7	Byte count	Number of bytes to read and return to caller

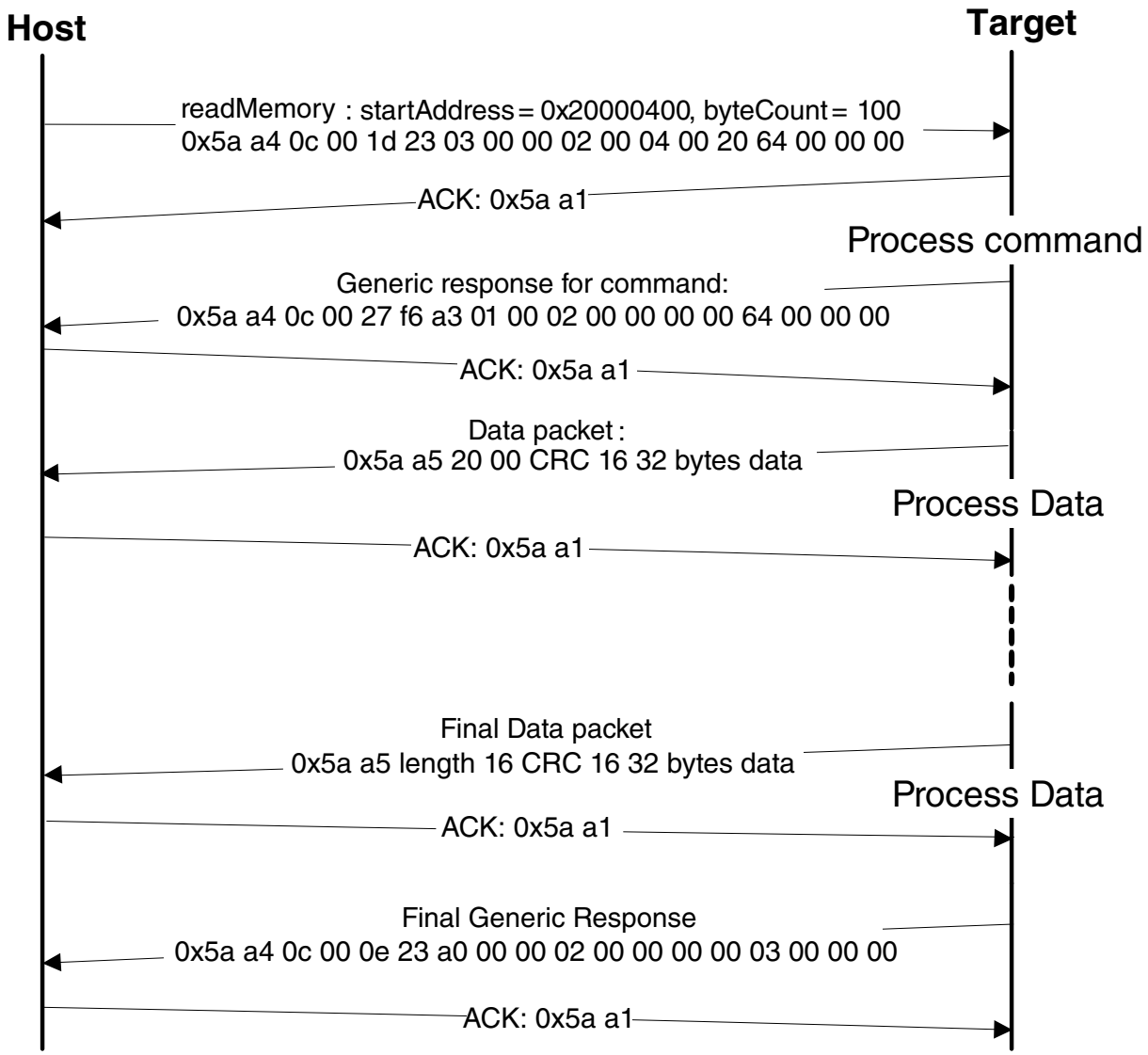


Figure 13-13. Command sequence for read memory

ReadMemory	Parameter	Value
Framing packet	Start byte	0x5A0xA4,
	packetType	kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x1D 0x23
Command packet	commandTag	0x03 - readMemory
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	startAddress	0x20000400
	byteCount	0x00000064

Data Phase: The ReadMemory command has a data phase. Since the target (Kinetis Flashloader) works in slave mode, the host need pull data packets until the number of bytes of data specified in the byteCount parameter of ReadMemory command are received by host.

Response: The target (Kinetis Flashloader) will return a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command, or set to an appropriate error status code.

13.2.6.8 FlashSecurityDisable command

The FlashSecurityDisable command performs the flash security disable operation, by comparing the 8-byte backdoor key (provided in the command) against the backdoor key stored in the flash configuration field (at address 0x400 in the flash).

The backdoor low and high words are the only parameters required for FlashSecurityDisable command.

Table 13-29. Parameters for FlashSecurityDisable Command

Byte #	Command
0 - 3	Backdoor key low word
4 - 7	Backdoor key high word

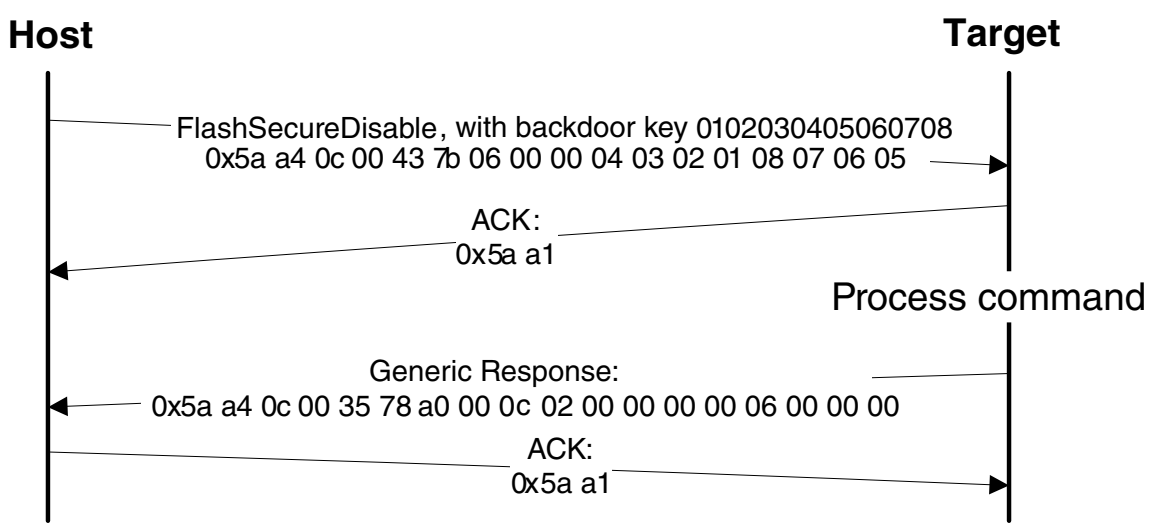


Figure 13-14. Protocol Sequence for FlashSecurityDisable Command

Table 13-30. FlashSecurityDisable Command Packet Format (Example)

FlashSecurityDisable	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x43 0x7B
Command packet	commandTag	0x06 - FlashSecurityDisable
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	Backdoorkey_low	0x04 0x03 0x02 0x01
	Backdoorkey_high	0x08 0x07 0x06 0x05

The FlashSecurityDisable command has no data phase.

Response: The target (Kinetis Flashloader) will return a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command, or set to an appropriate error status code.

13.2.6.9 Execute command

The execute command results in the flashloader setting the program counter to the code at the provided jump address, R0 to the provided argument, and a Stack pointer to the provided stack pointer address. Prior to the jump, the system is returned to the reset state.

The Jump address, function argument pointer, and stack pointer are the parameters required for the Execute command.

Table 13-31. Parameters for Execute Command

Byte #	Command
0 - 3	Jump address
4 - 7	Argument word
8 - 11	Stack pointer address

The Execute command has no data phase.

Response: Before executing the Execute command, the target (Kinetis Flashloader) will validate the parameters and return a GenericResponse packet with a status code either set to kStatus_Success or an appropriate error status code.

13.2.6.10 Reset command

The Reset command will result in flashloader resetting the chip.

The Reset command requires no parameters.

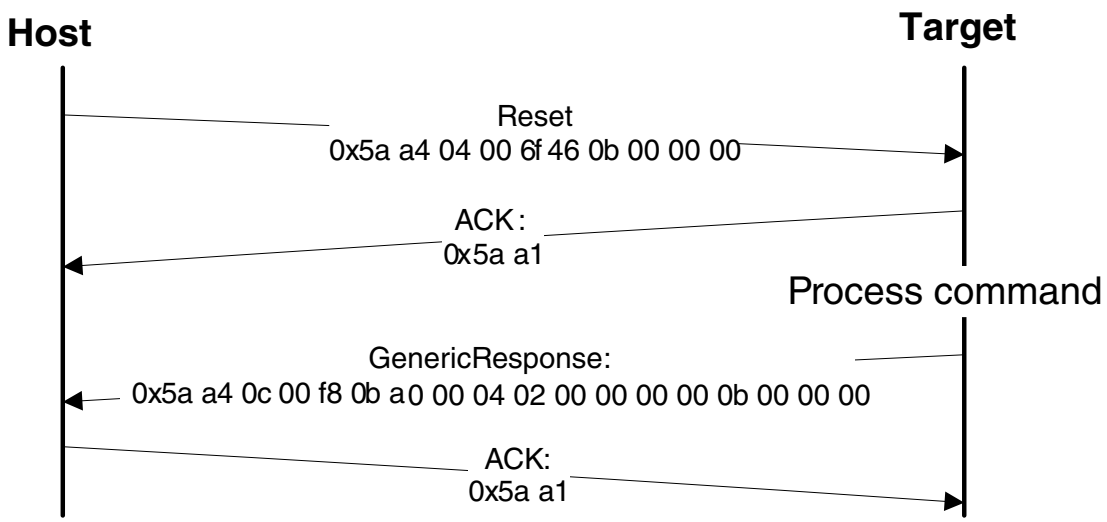


Figure 13-15. Protocol Sequence for Reset Command

Table 13-32. Reset Command Packet Format (Example)

Reset	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x04 0x00
	crc16	0x6F 0x46
Command packet	commandTag	0x0B - reset
	flags	0x00
	reserved	0x00
	parameterCount	0x00

The Reset command has no data phase.

Response: The target (Kinetis Flashloader) will return a GenericResponse packet with status code set to kStatus_Success, before resetting the chip.

13.3 Peripherals Supported

This section describes the peripherals supported by the Kinetis Flashloader.

13.3.1 I2C Peripheral

The Kinetis Flashloader supports loading data into flash via the I2C peripheral, where the I2C peripheral serves as the I2C slave. A 7-bit slave address is used during the transfer.

The Kinetis Flashloader uses 0x10 as the I2C slave address, and supports 400 kbps as the I2C baud rate.

Because the I2C peripheral serves as an I2C slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

- An incoming packet is sent by the host with a selected I2C slave address and the direction bit is set as write.
- An outgoing packet is read by the host with a selected I2C slave address and the direction bit is set as read.
- 0x00 will be sent as the response to host if the target is busy with processing or preparing data.

The following flow charts demonstrate the communication flow of how the host reads ping packet, ACK and response from the target.

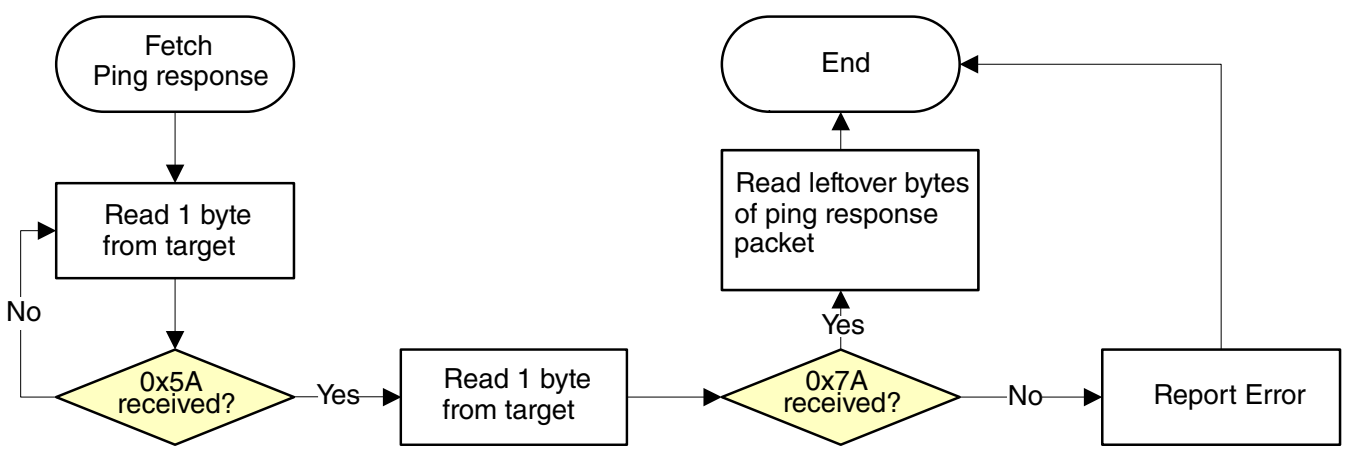


Figure 13-16. Host reads ping response from target via I2C

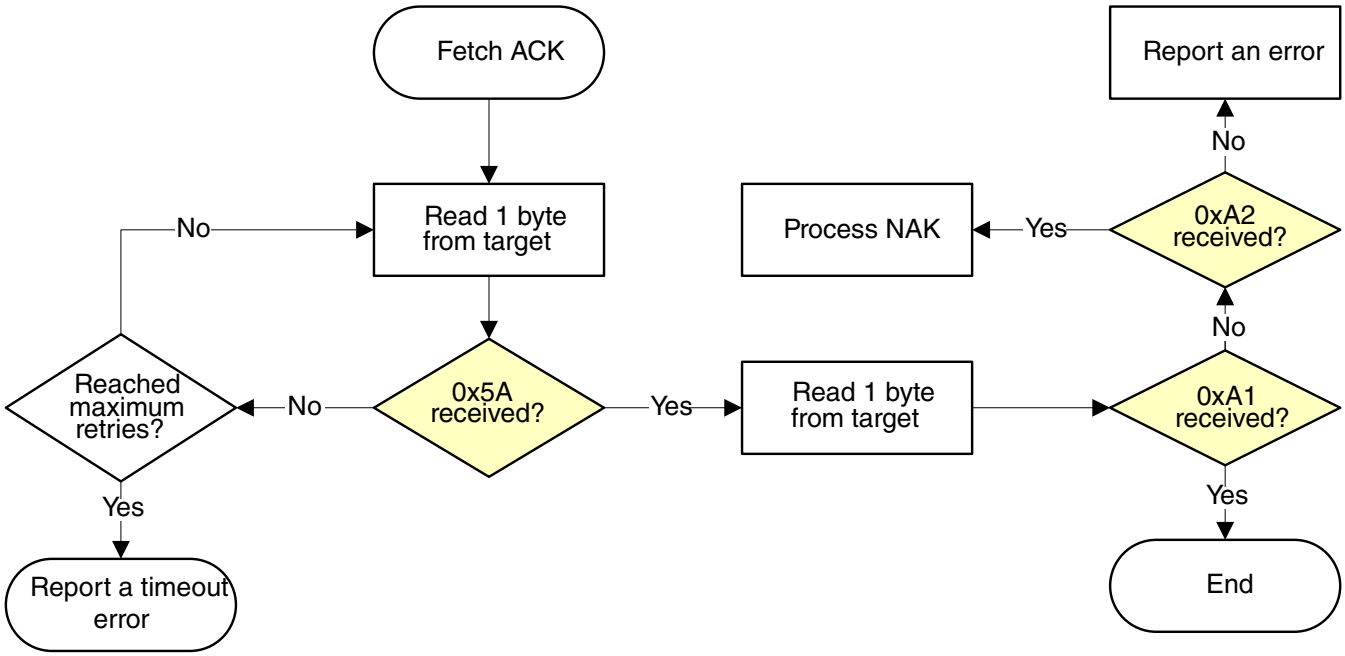


Figure 13-17. Host reads ACK packet from target via I2C

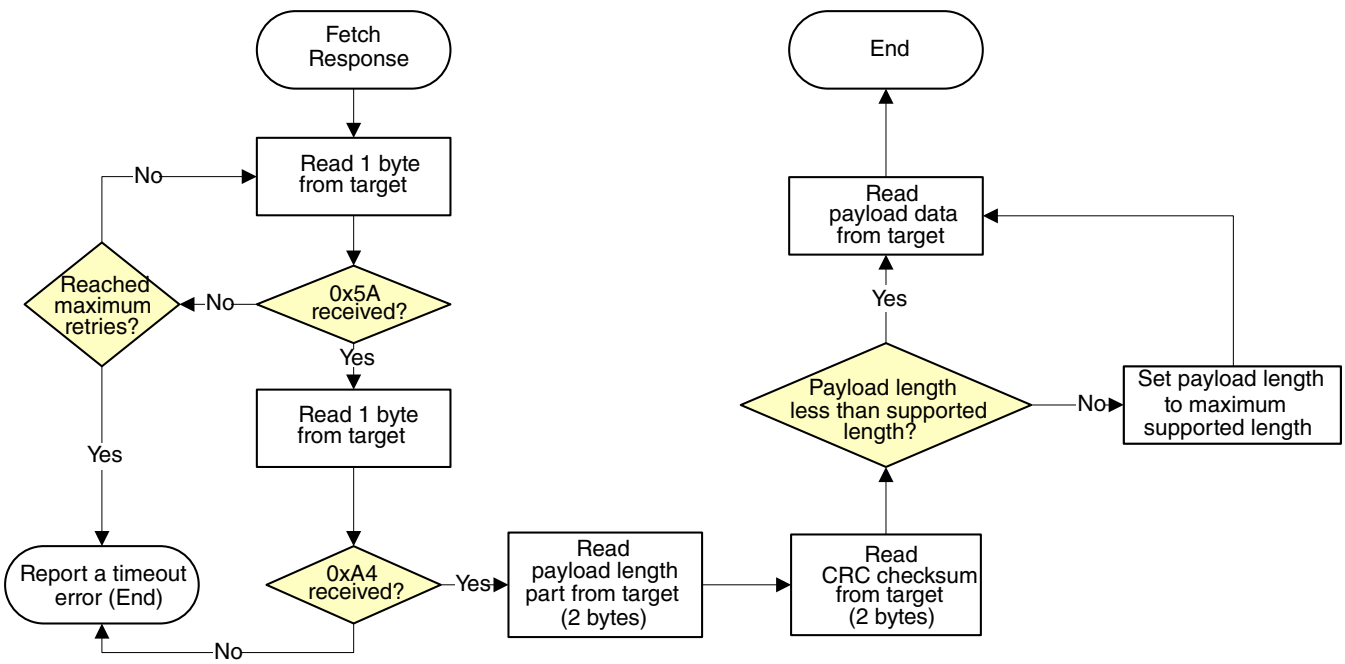


Figure 13-18. Host reads response from target via I2C

13.3.2 SPI Peripheral

The Kinetis Flashloader supports loading data into flash via the SPI peripheral, where the SPI peripheral serves as a SPI slave.

The Kinetis Flashloader supports 400 kbps as the SPI baud rate.

The SPI peripheral uses the following bus attributes:
- Clock Phase = 1 (Second Edge)
- Clock Polarity = 1 (Active Low)

Because the SPI peripheral serves as a SPI slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

The transfer on SPI is slightly different from I2C:
- Host will receive 1 byte after it sends out any byte.
- Received bytes should be ignored when host is sending out bytes to target
- Host starts reading bytes by sending 0x00s to target
- The byte 0x00 will be sent as response to host if target is under the following conditions:
  - Processing incoming packet
  - Preparing outgoing data
  - Received invalid data

The SPI bus configuration is:
- Phase = 1; data is sampled on rising edges
- Polarity = 1; idle is high
- MSB is transmitted first

For any transfer where the target does not have actual data to send, the target (slave) is responsible for ensuring that 0x00 bytes will be returned to the host (master). The host uses framing packets to identify real data and not "dummy" 0x00 bytes (which do not have framing packets).

The following flowcharts demonstrate how the host reads a ping response, an ACK and a command response from target via SPI.

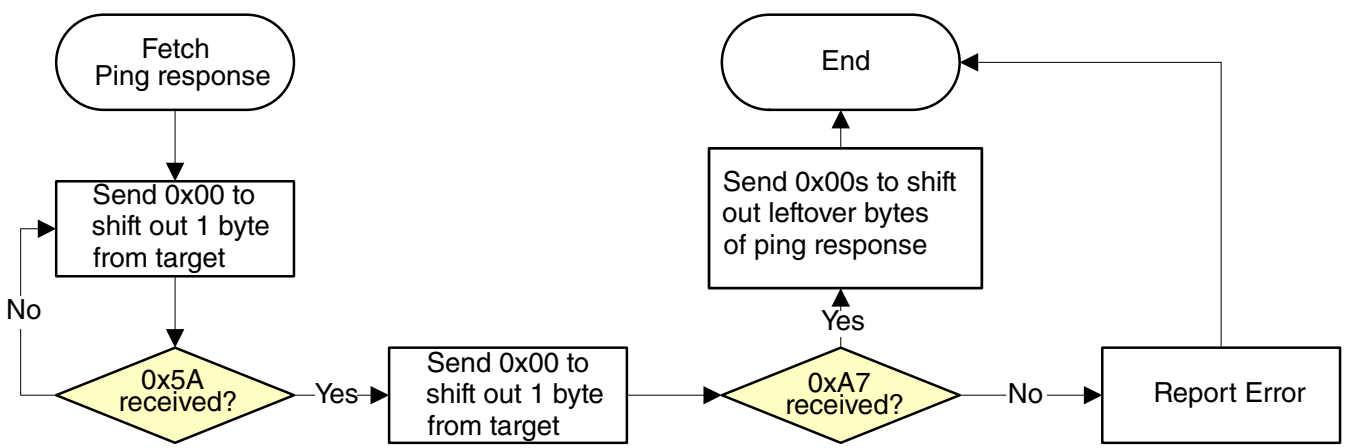


Figure 13-19. Host reads ping packet from target via SPI

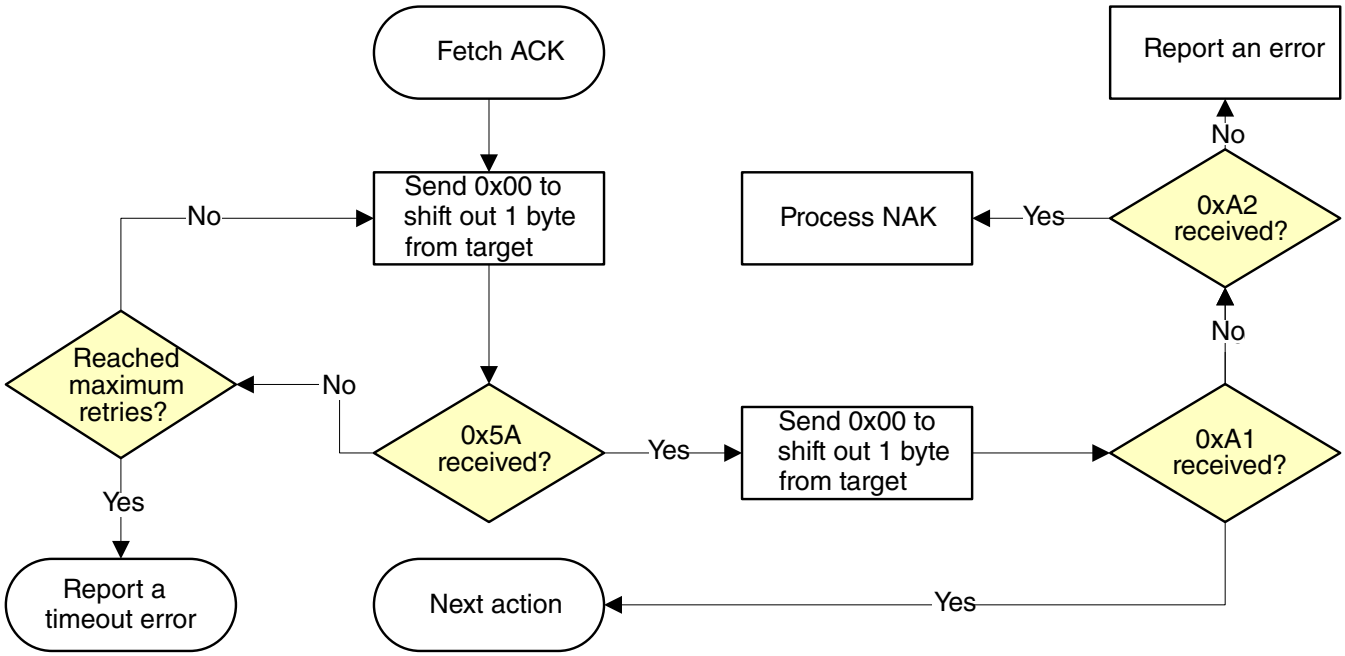


Figure 13-20. Host reads ACK from target via SPI

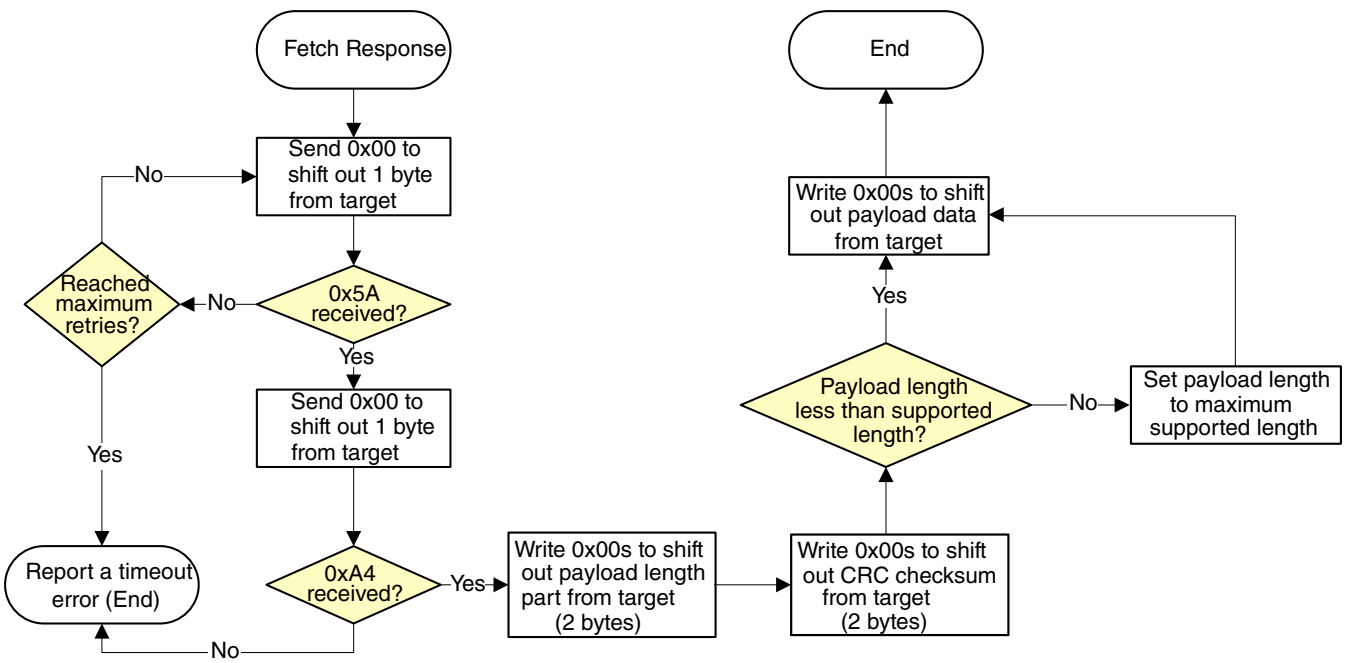


Figure 13-21. Host reads response from target via SPI

13.3.3 UART Peripheral

The Kinetis Flashloader integrates an autobaud detection algorithm for the UART peripheral, thereby providing flexible baud rate choices.

Autobaud feature: If UART*n* is used to connect to the flashloader, then the UART*n*_RX pin must be kept high and not left floating during the detection phase in order to comply with the autobaud detection algorithm. After the flashloader detects the ping packet (0x5A 0xA6) on UART*n*_RX, the flashloader firmware executes the autobaud sequence. If the baudrate is successfully detected, then the flashloader will send a ping packet response [(0x5A 0xA7), protocol version (4 bytes), protocol version options (2 bytes) and crc16 (2 bytes)] at the detected baudrate. The Kinetis Flashloader then enters a loop, waiting for flashloader commands via the UART peripheral.

NOTE

- The autobaud feature requires a ping packet with a higher accuracy (+/-3%), or the ping packet will be ignored as noise.
- The data bytes of the ping packet must be sent continuously (with no more than 80 ms between bytes) in a fixed UART transmission mode (8-bit data, no parity bit and 1 stop bit). If the bytes of the ping packet are sent one-by-one with more than 80 ms delay between them, then the autobaud detection algorithm may calculate an incorrect baud rate. In this case, the autobaud detection state machine should be reset.

Supported baud rates: The baud rate is closely related to the MCU core and system clock frequencies. Typical baud rates supported are 9600, 19200, 38400, 57600, and 115200.

Packet transfer: After autobaud detection succeeds, flashloader communications can take place over the UART peripheral. The following flow charts show:

- How the host detects an ACK from the target
- How the host detects a ping response from the target
- How the host detects a command response from the target

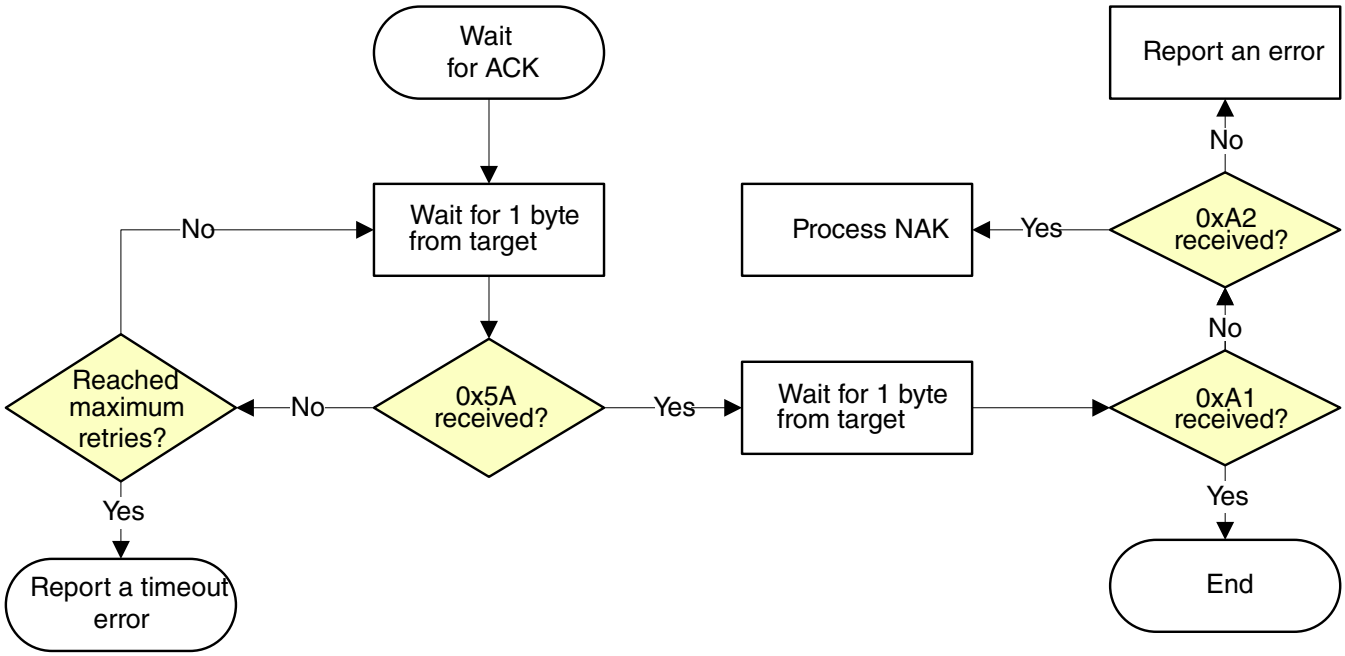


Figure 13-22. Host reads an ACK from target via UART

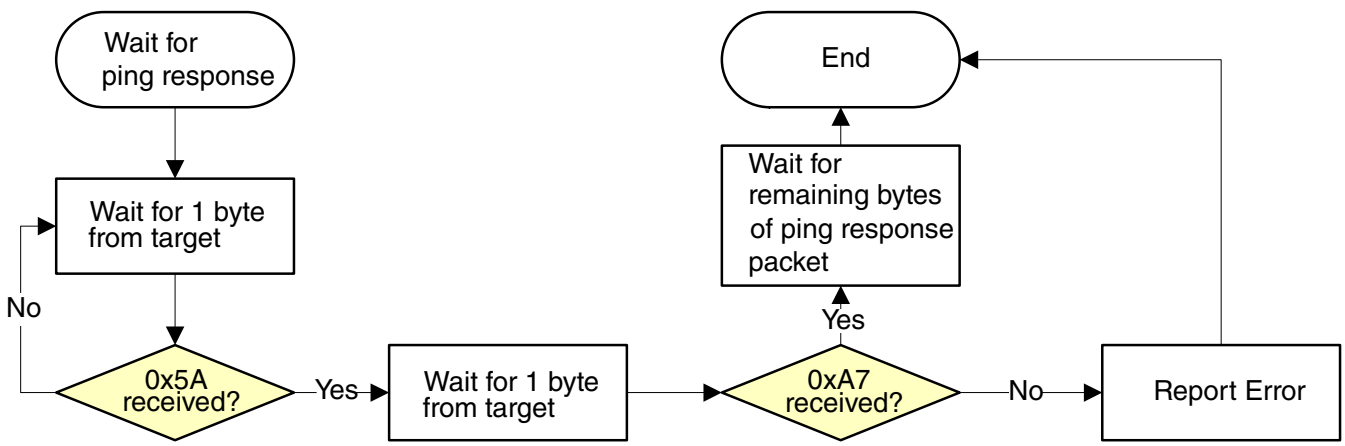


Figure 13-23. Host reads a ping response from target via UART

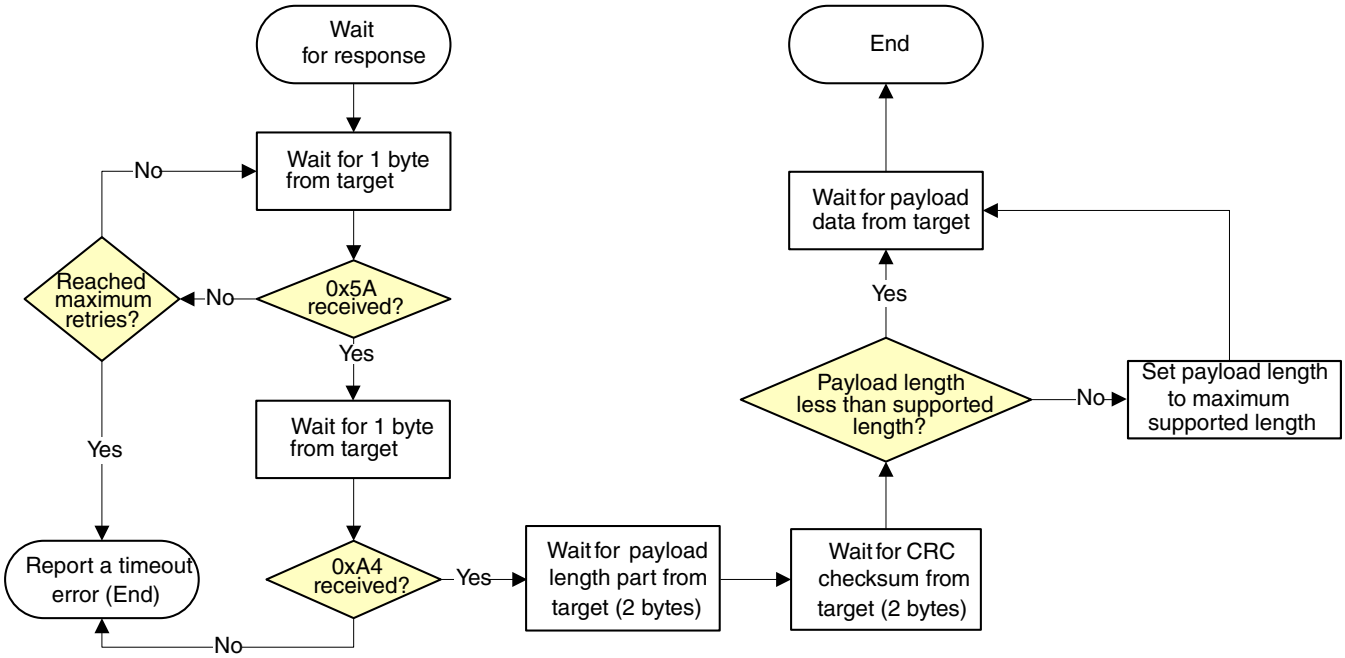


Figure 13-24. Host reads a command response from target via UART

13.3.4 CAN (or FlexCAN) Peripheral

The Kinetis Flashloader supports loading data into flash via the FlexCAN peripheral. Transfers to FlexCAN are supported at several predefined speeds:

- 125 kHz
- 250 kHz
- 500 kHz
- 1 MHz (the default transfer rate)

The host application must use one of the several supported speeds for FlexCAN. In Flashloader, it supports automatic speed detection within supported speeds. The Flashloader will enter the listen mode in the beginning with the initial speed (default speed 1 MHz). Once the host sends a ping to a specific node, it will generate traffic on the FlexCAN bus. Because the Flashloader is in a listen mode, it will be able to check if the local node speed is correct, by detecting errors.

- If there is an error, then some transfers may not be at the right speed.
- The Flashloader will change the speed setting and check again.
- If there is no error, it means that the transfer speed is correct, and it changes the settings back to normal receiving mode, to see if there is a package for this node.
- The host side should also have reasonable time tolerance during the automatic speed detection period. If there is a timeout, it means that there is no response from the specific node, or there is a real error (and it should report the error to the application).

The following flowcharts show how the host reads a ping packet, ACK and response from the target.

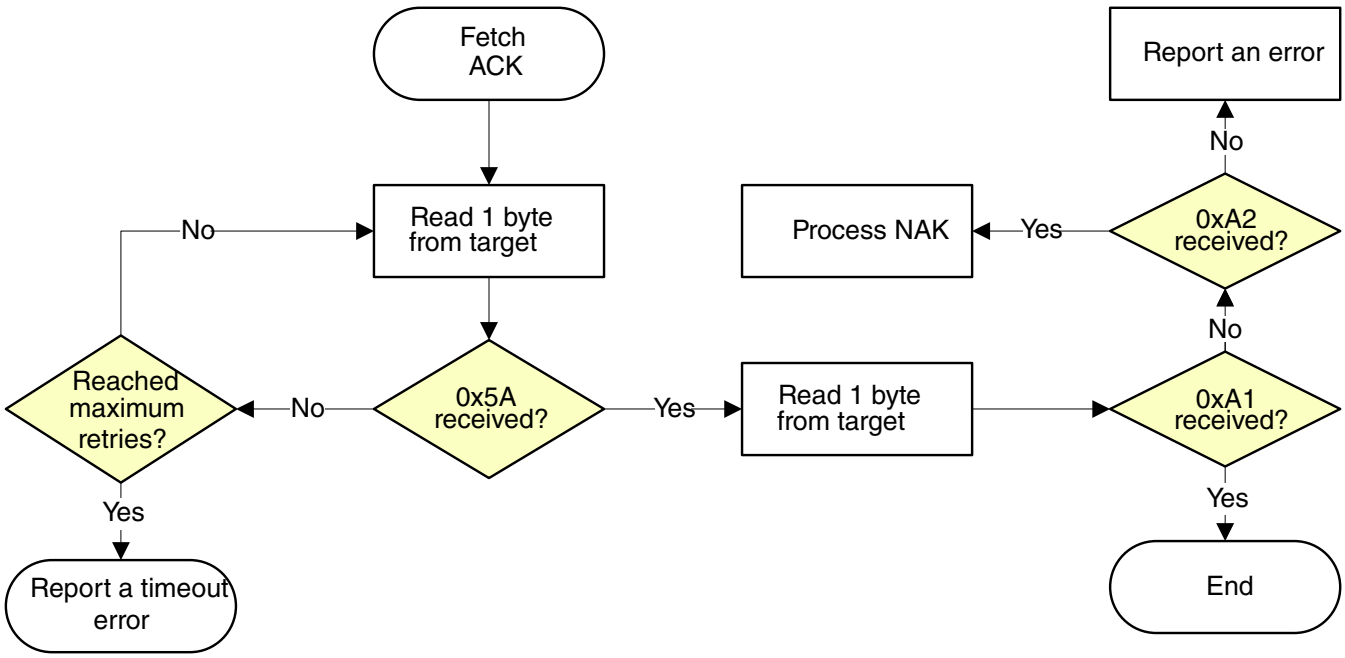


Figure 13-25. Host reads an ACK from target via FlexCAN

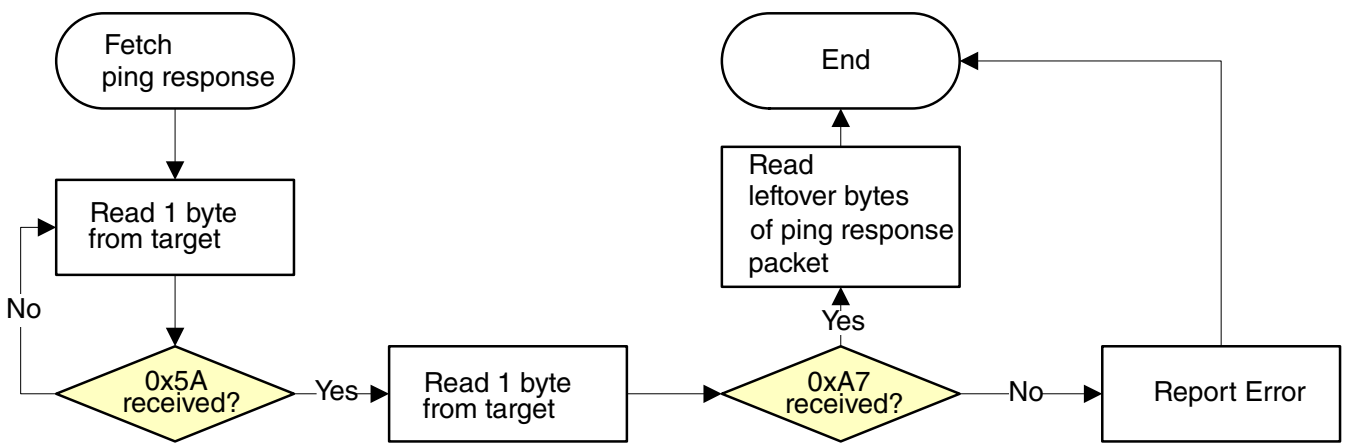


Figure 13-26. Host reads a ping response from target via FlexCAN

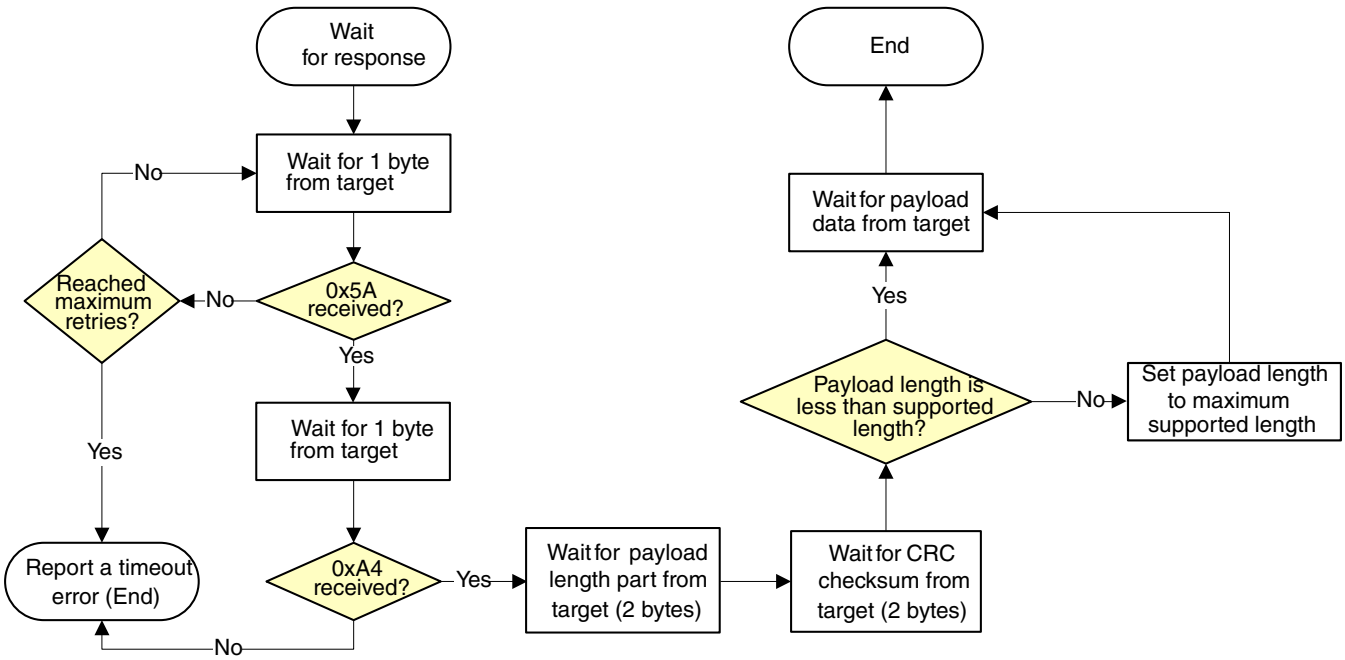


Figure 13-27. Host reads a command response from target via FlexCAN

13.4 Get/SetProperty Command Properties

This section lists the properties of the GetProperty and SetProperty commands.

Table 13-33. Properties used by Get/SetProperty Commands, sorted by Value

Property	Writable	Tag Value	Size	Description
CurrentVersion	No	01h	4	Current flashloader version.
AvailablePeripherals	No	02h	4	The set of peripherals supported on this chip.
FlashStartAddress	No	03h	4	Start address of program flash.
FlashSizeInBytes	No	04h	4	Size in bytes of program flash.
FlashSectorSize	No	05h	4	The size in bytes of one sector of program flash. This is the minimum erase size.
FlashBlockCount	No	06h	4	Number of blocks in the flash array.
AvailableCommands	No	07h	4	The set of commands supported by the flashloader.
VerifyWrites	Yes	0Ah	4	Controls whether the flashloader will verify writes to flash. VerifyWrites feature is enabled by default. 0 - No verification is done. 1 - Enable verification.
MaxPacketSize	No	0Bh	4	Maximum supported packet size for the currently active peripheral interface.

Table continues on the next page...

Table 13-33. Properties used by Get/SetProperty Commands, sorted by Value (continued)

Property	Writable	Tag Value	Size	Description
ReservedRegions	No	0Ch	16	List of memory regions reserved by the flashloader. Returned as value pairs (<start-address-of-region>, <end-address-of-region>). • If HasDataPhase flag is not set, then the Response packet parameter count indicates the number of pairs. • If HasDataPhase flag is set, then the second parameter is the number of bytes in the data phase.
RAMStartAddress	No	0Eh	4	Start address of RAM segment. The first parameter to GetProperty command identifies the segment. See the device specific memory map for number of RAM segments the device contains.
RAMSizeInBytes	No	0Fh	4	Size in bytes of RAM segment. The first parameter to GetProperty command identifies the segment. See the device specific memory map for number of RAM segments the device contains.
SystemDeviceId	No	10h	4	Value of the Kinetis System Device Identification register.
FlashSecurityState	No	11h	4	Indicates whether Flash security is enabled 0 - Flash security is disabled 1 - Flash security is enabled
UniqueDeviceId	No	12h	16	Unique device identification, value of Kinetis Unique Identification registers (16 for K series devices, 12 for KL series devices)
FlashReadMargin	Yes	16h	4	The margin level setting for flash erase and program verify commands. 0 = Normal 1 = User (default) 2 = Factory
TargetVersion	No	18h	4	SoC target build version number

13.4.1 Property Definitions

Get/Set property definitions are provided in this section.

13.4.1.1 CurrentVersion Property

The value of this property is a 4-byte structure containing the current version of the flashloader.

Table 13-34. Fields of CurrentVersion property:

Bits	[31:24]	[23:16]	[15:8]	[7:0]
Field	Name = 'K' (0x4B)	Major version	Minor version	Bugfix version

13.4.1.2 AvailablePeripherals Property

The value of this property is a bitfield that lists the peripherals supported by the flashloader and the hardware on which it is running.

Table 13-35. Peripheral bits:

Bit	[31:7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Peripheral	Reserved	Reserved	Reserved	Reserved	CAN Slave	SPI Slave	I2C Slave	UART

If the peripheral is available, then the corresponding bit will be set in the property value. All reserved bits must be set to 0.

13.4.1.3 AvailableCommands Property

This property value is a bitfield with set bits indicating the commands enabled in the flashloader. Only commands that can be sent from the host to the target are listed in the bitfield. Response commands such as GenericResponse are excluded.

The bit number that identifies whether a command is present is the command's tag value minus 1. 1 is subtracted from the command tag because the lowest command tag value is 0x01. To get the bit mask for a given command, use this expression:

```
mask = 1 << (tag - 1)
```

Table 13-36. Command bits:

Bit	[31:18]	[17]	[16]	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]

Table continues on the next page...

Table 13-36. Command bits: (continued)

Command	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	SetProperty	Reset	Reserved	Execute	Reserved	GetProperty	FlashSecurityDisable	FillMemory	WriteMemory	ReadMemory	FlashEraseRegion	FlashEraseAll

13.5 Kinetis Flashloader Status Error Codes

This section describes the status error codes that the Kinetis Flashloader returns to the host.

Table 13-37. Kinetis Flashloader Status Error Codes, sorted by Value

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	Operation failed with a generic error.
kStatus_ReadOnly	2	Requested value cannot be changed because it is read-only.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_FlashSizeError	100	Not used.
kStatus_FlashAlignmentError	101	Address or length does not meet required alignment.
kStatus_FlashAddressError	102	Address or length is outside addressable memory.
kStatus_FlashAccessError	103	The FTFA_FSTAT[ACCERR] bit is set.
kStatus_FlashProtectionViolation	104	The FTFA_FSTAT[FPVIOL] bit is set.
kStatus_FlashCommandFailure	105	The FTFA_FSTAT[MGSTAT0] bit is set.
kStatus_FlashUnknownProperty	106	Unknown Flash property.
kStatus_FlashEraseKeyError	107	The key provided does not match the programmed flash key.
kStatus_FlashRegionExecuteOnly	108	The area of flash is protected as execute only.
kStatus_I2C_SlaveTxUnderrun	200	I2C Slave TX Underrun error.
kStatus_I2C_SlaveRxOverrun	201	I2C Slave RX Overrun error.
kStatus_I2C_AribtrationLost	202	I2C Arbitration Lost error.
kStatus_SPI_SlaveTxUnderrun	300	SPI Slave TX Underrun error.
kStatus_SPI_SlaveRxOverrun	301	SPI Slave RX Overrun error.
kStatus_SPI_Timeout	302	SPI tranfser timed out.

Table continues on the next page...

Table 13-37. Kinetis Flashloader Status Error Codes, sorted by Value (continued)

Error Code	Value	Description
kStatus_SPI_Busy	303	SPI instance is already busy performing a transfer.
kStatus_SPI_NoTransferInProgress	304	Attempt to abort a transfer when no transfer was in progress.
kStatus_UnknownCommand	10000	The requested command value is undefined.
kStatus_SecurityViolation	10001	Command is disallowed because flash security is enabled.
kStatus_AbortDataPhase	10002	Abort the data phase early.
kStatusMemoryRangeInvalid	10200	Memory range conflicts with a protected region.
kStatus_UnknownProperty	10300	The requested property value is undefined.
kStatus_ReadOnlyProperty	10301	The requested property value cannot be written.
kStatus_InvalidPropertyValue	10302	The specified property value is invalid.
kStatus_AppCrcCheckPassed	10400	CRC check is valid and passed.
kStatus_AppCrcCheckFailed	10401	CRC check is valid but failed.
kStatus_AppCrcCheckInactive	10402	CRC check is inactive.
kStatus_AppCrcCheckInvalid	10403	CRC check is invalid, because the BCA is invalid or the CRC parameters are unset (all 0xFF bytes).
kStatus_AppCrcCheckOutOfRange	10404	CRC check is valid but addresses are out of range.

Chapter 14
System Mode Controller (SMC)

14.1 Introduction

NOTE

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

The System Mode Controller (SMC) is responsible for sequencing the system into and out of all low-power Stop and Run modes.

Specifically, it monitors events to trigger transitions between power modes while controlling the power, clocks, and memories of the system to achieve the power consumption and functionality of that mode.

This chapter describes all the available low-power modes, the sequence followed to enter/exit each mode, and the functionality available while in each of the modes.

The SMC is able to function during even the deepest low power modes.

See AN4503: Power Management for Kinetis MCUs for further details on using the SMC.

14.2 Modes of operation

The ARM CPU has three primary modes of operation:

- Run
- Sleep
- Deep Sleep

The WFI or WFE instruction is used to invoke Sleep and Deep Sleep modes. Run, Wait, and Stop are the common terms used for the primary operating modes of Kinetis microcontrollers.

The following table shows the translation between the ARM CPU modes and the Kinetis MCU power modes.

ARM CPU mode	MCU mode
Sleep	Wait
Deep Sleep	Stop

Accordingly, the ARM CPU documentation refers to sleep and deep sleep, while the Kinetis MCU documentation normally uses wait and stop.

In addition, Kinetis MCUs also augment Stop, Wait, and Run modes in a number of ways. The power management controller (PMC) contains a run and a stop mode regulator. Run regulation is used in normal run, wait and stop modes. Stop mode regulation is used during all very low power and low leakage modes. During stop mode regulation, the bus frequencies are limited in the very low power modes.

The SMC provides the user with multiple power options. The Very Low Power Run (VLPR) mode can drastically reduce run time power when maximum bus frequency is not required to handle the application needs. From Normal Run mode, the Run Mode (RUNM) field can be modified to change the MCU into VLPR mode when limited frequency is sufficient for the application. From VLPR mode, a corresponding wait (VLPW) and stop (VLPS) mode can be entered.

Depending on the needs of the user application, a variety of stop modes are available that allow the state retention, partial power down or full power down of certain logic and/or memory. I/O states are held in all modes of operation. Several registers are used to configure the various modes of operation for the device.

The following table describes the power modes available for the device.

Table 14-1. Power modes

Mode	Description
RUN	The MCU can be run at full speed and the internal supply is fully regulated, that is, in run regulation. This mode is also referred to as Normal Run mode.
WAIT	The core clock is gated off. The system clock continues to operate. Bus clocks, if enabled, continue to operate. Run regulation is maintained.
STOP	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid.
VLPR	The core, system, bus, and flash clock maximum frequencies are restricted in this mode. See the Power Management chapter for details about the maximum allowable frequencies.
VLPW	The core clock is gated off. The system, bus, and flash clocks continue to operate, although their maximum frequency is restricted. See the Power Management chapter for details on the maximum allowable frequencies.
VLPS	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid.

Table continues on the next page...

Table 14-1. Power modes (continued)

Mode	Description
VLLS3	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic. All system RAM contents are retained and I/O states are held. Internal logic states are not retained.
VLLS1	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic and all system RAM. I/O states are held. Internal logic states are not retained.
VLLS0	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic and all system RAM. I/O states are held. Internal logic states are not retained. The 1kHz LPO clock is disabled and the power on reset (POR) circuit can be optionally enabled using STOPCTRL[PORPO].

14.3 Memory map and register descriptions

Information about the registers related to the system mode controller can be found here.

Different SMC registers reset on different reset types. Each register's description provides details. For more information about the types of reset on this chip, refer to the Reset section details.

NOTE

The SMC registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

NOTE

Before executing the WFI instruction, the last register written to must be read back. This ensures that all register writes associated with setting up the low power mode being entered have completed before the MCU enters the low power mode. Failure to do this may result in the low power mode not being entered correctly.

SMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_E000	Power Mode Protection register (SMC_PMPROT)	8	R/W	00h	14.3.1/292
4007_E001	Power Mode Control register (SMC_PMCTRL)	8	R/W	00h	14.3.2/293

Table continues on the next page...

SMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_E002	Stop Control Register (SMC_STOPCTRL)	8	R/W	03h	14.3.3/294
4007_E003	Power Mode Status register (SMC_PMSTAT)	8	R	01h	14.3.4/296

14.3.1 Power Mode Protection register (SMC_PMPROT)

This register provides protection for entry into any low-power run or stop mode. The enabling of the low-power run or stop mode occurs by configuring the Power Mode Control register (PMCTRL).

The PMPROT register can be written only once after any system reset.

If the MCU is configured for a disallowed or reserved power mode, the MCU remains in its current power mode. For example, if the MCU is in normal RUN mode and AVLP is 0, an attempt to enter VLPR mode using PMCTRL[RUNM] is blocked and PMCTRL[RUNM] remains 00b, indicating the MCU is still in Normal Run mode.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Reset section details for more information.

Address: 4007_E000h base + 0h offset = 4007_E000h

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

SMC_PMPROT field descriptions

Field	Description
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 read-only field is reserved and always has the value 0.
5 AVLP	Allow Very-Low-Power Modes Provided the appropriate control bits are set up in PMCTRL, this write-once field allows the MCU to enter any very-low-power mode (VLPR, VLPW, and VLPS). 0 VLPR, VLPW, and VLPS are not allowed. 1 VLPR, VLPW, and VLPS are allowed.

Table continues on the next page...

SMC_PMPROT field descriptions (continued)

Field	Description
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.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 AVLLS	Allow Very-Low-Leakage Stop Mode Provided the appropriate control bits are set up in PMCTRL, this write once bit allows the MCU to enter any very-low-leakage stop mode (VLLSx). 0 Any VLLSx mode is not allowed 1 Any VLLSx mode is allowed
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

14.3.2 Power Mode Control register (SMC_PMCTRL)

The PMCTRL register controls entry into low-power Run and Stop modes, provided that the selected power mode is allowed via an appropriate setting of the protection (PMPROT) register.

NOTE

This register is reset on Chip POR not VLLS and by reset types that trigger Chip POR not VLLS. It is unaffected by reset types that do not trigger Chip POR not VLLS. See the Reset section details for more information.

Address: 4007_E000h base + 1h offset = 4007_E001h

Bit	7	6	5	4	3	2	1	0
Read	Reserved	RUNM		0	STOPA	STOPM		
Write								
Reset	0	0	0	0	0	0	0	0

SMC_PMCTRL field descriptions

Field	Description
7 Reserved	This field is reserved. This bit is reserved for future expansion. Software should write 0 to this bit to maintain compatibility.
6–5 RUNM	Run Mode Control When written, causes entry into the selected run mode. Writes to this field are blocked if the protection level has not been enabled using the PMPROT register.

Table continues on the next page...

SMC_PMCTRL field descriptions (continued)

Field	Description
	NOTE: RUNM may be set to VLPR only when PMSTAT=RUN. After being written to VLPR, RUNM should not be written back to RUN until PMSTAT=VLPR. 00 Normal Run mode (RUN) 01 Reserved 10 Very-Low-Power Run mode (VLPR) 11 Reserved
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 STOPA	Stop Aborted When set, this read-only status bit indicates an interrupt or reset occured during the previous stop mode entry sequence, preventing the system from entering that mode. This field is cleared by hardware at the beginning of any stop mode entry sequence and is set if the sequence was aborted. 0 The previous stop mode entry was successful. 1 The previous stop mode entry was aborted.
STOPM	Stop Mode Control When written, controls entry into the selected stop mode when Sleep-Now or Sleep-On-Exit mode is entered with SLEEPDEEP=1 . Writes to this field are blocked if the protection level has not been enabled using the PMPROT register. After any system reset, this field is cleared by hardware on any successful write to the PMPROT register. **NOTE:** When set to VLLSx, the VLLSM field in the STOPCTRL register is used to further select the particular VLLS submode which will be entered. **NOTE:** When set to STOP, the PSTOPO bits in the STOPCTRL register can be used to select a Partial Stop mode if desired. 000 Normal Stop (STOP) 001 Reserved 010 Very-Low-Power Stop (VLPS) 011 Reserved 100 Very-Low-Leakage Stop (VLLSx) 101 Reserved 110 Reseved 111 Reserved

14.3.3 Stop Control Register (SMC_STOPCTRL)

The STOPCTRL register provides various control bits allowing the user to fine tune power consumption during the stop mode selected by the STOPM field.

NOTE

This register is reset on Chip POR not VLLS and by reset types that trigger Chip POR not VLLS. It is unaffected by reset types

that do not trigger Chip POR not VLLS. See the Reset section details for more information.

Address: 4007_E000h base + 2h offset = 4007_E002h

Bit	7	6	5	4	3	2	1	0
Read	PSTOPO		PORPO	0	Reserved	VLLSM		
Write								
Reset	0	0	0	0	0	0	1	1

SMC_STOPCTRL field descriptions

Field	Description
7–6 PSTOPO	Partial Stop Option These bits control whether a Partial Stop mode is entered when STOPM=STOP. When entering a Partial Stop mode from RUN (or VLPR) mode, the PMC, MCG and flash remain fully powered, allowing the device to wakeup almost instantaneously at the expense of higher power consumption. In PSTOP2, only system clocks are gated allowing peripherals running on bus clock to remain fully functional. In PSTOP1, both system and bus clocks are gated. 00 STOP - Normal Stop mode 01 PSTOP1 - Partial Stop with both system and bus clocks disabled 10 PSTOP2 - Partial Stop with system clock disabled and bus clock enabled 11 Reserved
5 PORPO	POR Power Option This bit controls whether the POR detect circuit is enabled in VLLS0 mode. 0 POR detect circuit is enabled in VLLS0 1 POR detect circuit is disabled in VLLS0
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 bit is reserved for future expansion. Software should write 0 to this bit to maintain compatibility.
VLLSM	VLLS Mode Control This field controls which VLLS sub-mode to enter if STOPM = VLLSx. 000 VLLS0 001 VLLS1 010 Reserved 011 VLLS3 100 Reserved 101 Reserved 110 Reserved 111 Reserved

14.3.4 Power Mode Status register (SMC_PMSTAT)

PMSTAT is a read-only, one-hot register which indicates the current power mode of the system.

NOTE

This register is reset on Chip POR not VLLS and by reset types that trigger Chip POR not VLLS. It is unaffected by reset types that do not trigger Chip POR not VLLS. See the Reset section details for more information.

Address: 4007_E000h base + 3h offset = 4007_E003h

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

SMC_PMSTAT field descriptions

Field	Description
PMSTAT	Power Mode Status **NOTE:** When debug is enabled, the PMSTAT will not update to STOP or VLPS **NOTE:** When a PSTOP mode is enabled, the PMSTAT will not update to STOP or VLPS 0000_0001 Current power mode is RUN. 0000_0010 Current power mode is STOP. 0000_0100 Current power mode is VLPR. 0000_1000 Current power mode is VLPW. 0001_0000 Current power mode is VLPS. 0010_0000 Reserved 0100_0000 Current power mode is VLLS. 1000_0000 Reserved

14.4 Functional description

14.4.1 Power mode transitions

The following figure shows the power mode state transitions available on the chip. Any reset always brings the MCU back to the normal RUN state.

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

Table 14-2. Power mode transition triggers

Transition #	From	To	Trigger conditions
1	RUN	WAIT	Sleep-now or sleep-on-exit modes entered with SLEEPDEEP clear, controlled in System Control Register in ARM core. See note.[1]
	WAIT	RUN	Interrupt or Reset
2	RUN	STOP	PMCTRL[RUNM]=00, PMCTRL[STOPM]=000[2] Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. See note.[1]
	STOP	RUN	Interrupt or Reset
3	RUN	VLPR	The core, system, bus and flash clock frequencies and MCG clocking mode are restricted in this mode. See the Power Management chapter for the maximum allowable frequencies and MCG modes supported. Set PMPROT[AVLP]=1, PMCTRL[RUNM]=10.
	VLPR	RUN	Set PMCTRL[RUNM]=00 or Reset.
4	VLPR	VLPW	Sleep-now or sleep-on-exit modes entered with SLEEPDEEP clear, which is controlled in System Control Register in ARM core. See note.[1]
	VLPW	VLPR	Interrupt
5	VLPW	RUN	Reset
6	VLPR	VLPS	PMCTRL[STOPM]=000[3] or 010, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. See note.[1]
	VLPS	VLPR	Interrupt **NOTE:** If VLPS was entered directly from RUN (transition #4), hardware forces exit back to RUN and does not allow a transition to VLPR.
7	RUN	VLPS	PMPROT[AVLP]=1, PMCTRL[STOPM]=010, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. See note.[1]
	VLPS	RUN	Interrupt and VLPS mode was entered directly from RUN or Reset

Table continues on the next page...

Table 14-2. Power mode transition triggers (continued)

Transition #	From	To	Trigger conditions
8	RUN	VLLSx	PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, STOPCTRL[LLSM]=x (VLLSx), Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.
	VLLSx	RUN	Wakeup from enabled LLWU input source or RESET pin
9	VLPR	VLLSx	PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, STOPCTRL[LLSM]=x (VLLSx), Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.

1. If debug is enabled, the core clock remains to support debug.
2. If PMCTRL[STOPM]=000 and STOPCTRL[PSTOPO]=01 or 10, then only a Partial Stop mode is entered instead of STOP
3. If PMCTRL[STOPM]=000 and STOPCTRL[PSTOPO]=00, then VLPS mode is entered instead of STOP. If PMCTRL[STOPM]=000 and STOPCTRL[PSTOPO]=01 or 10, then only a Partial Stop mode is entered instead of VLPS

14.4.2 Power mode entry/exit sequencing

When entering or exiting low-power modes, the system must conform to an orderly sequence to manage transitions safely.

The SMC manages the system's entry into and exit from all power modes. This diagram illustrates the connections of the SMC with other system components in the chip that are necessary to sequence the system through all power modes.

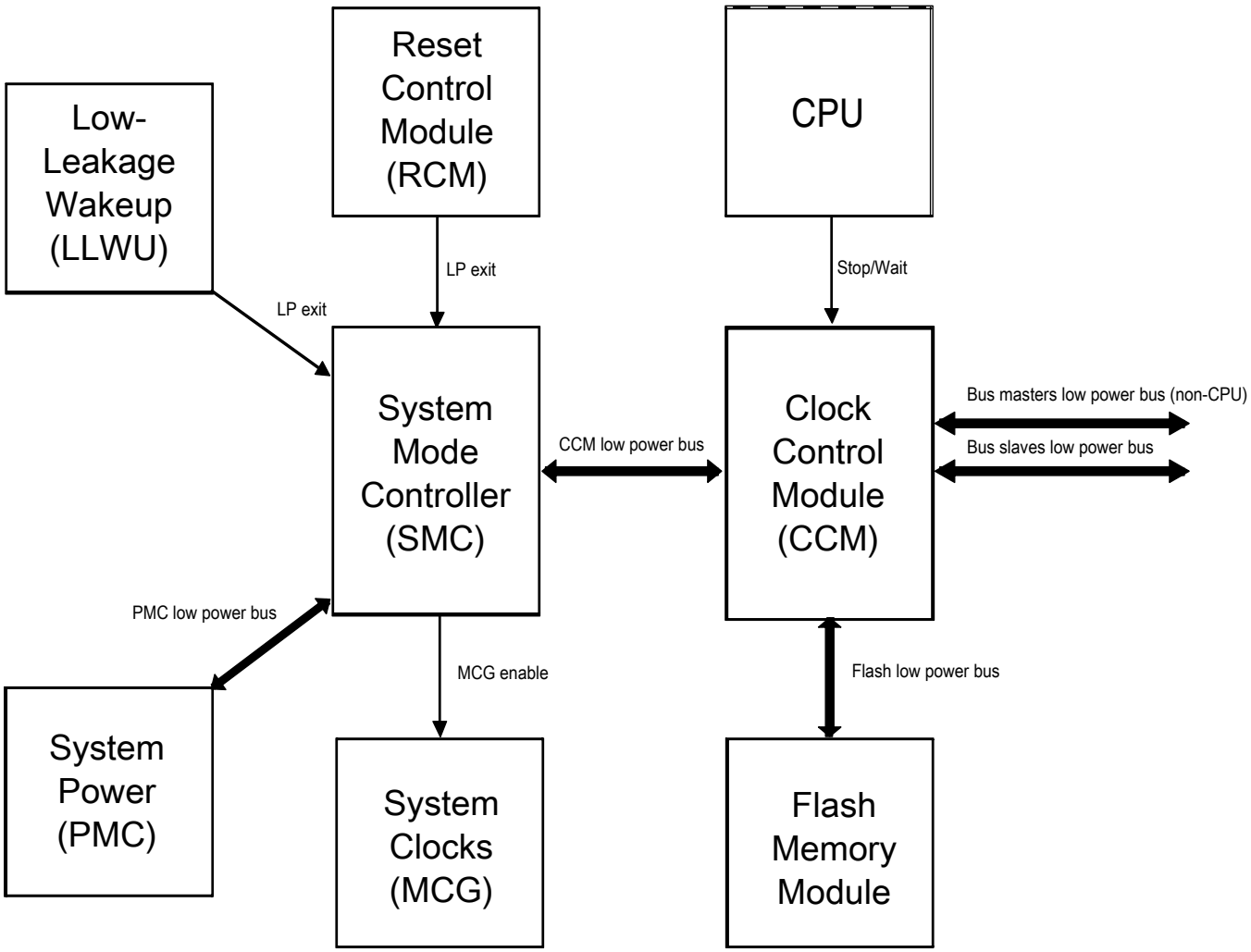


Figure 14-1. Low-power system components and connections

14.4.2.1 Stop mode entry sequence

Entry into a low-power stop mode (Stop, VLPS, VLLSx) is initiated by a CPU executing the WFI instruction. After the instruction is executed, the following sequence occurs:

1. The CPU clock is gated off immediately.
2. Requests are made to all non-CPU bus masters to enter Stop mode.
3. After all masters have acknowledged they are ready to enter Stop mode, requests are made to all bus slaves to enter Stop mode.
4. After all slaves have acknowledged they are ready to enter Stop mode, all system and bus clocks are gated off.
5. Clock generators are disabled in the MCG.
6. The on-chip regulator in the PMC and internal power switches are configured to meet the power consumption goals for the targeted low-power mode.

14.4.2.2 Stop mode exit sequence

Exit from a low-power stop mode is initiated either by a reset or an interrupt event. The following sequence then executes to restore the system to a run mode (RUN or VLPR):

1. The on-chip regulator in the PMC and internal power switches are restored.
2. Clock generators are enabled in the MCG.
3. System and bus clocks are enabled to all masters and slaves.
4. The CPU clock is enabled and the CPU begins servicing the reset or interrupt that initiated the exit from the low-power stop mode.

14.4.2.3 Aborted stop mode entry

If an interrupt or a reset occurs during a stop entry sequence, the SMC can abort the transition early and return to RUN mode without completely entering the stop mode. An aborted entry is possible only if the reset or interrupt occurs before the PMC begins the transition to stop mode regulation. After this point, the interrupt or reset is ignored until the PMC has completed its transition to stop mode regulation. When an aborted stop mode entry sequence occurs, SMC_PMCTRL[STOPA] is set to 1.

14.4.2.4 Transition to wait modes

For wait modes (WAIT and VLPW), the CPU clock is gated off while all other clocking continues, as in RUN and VLPR mode operation. Some modules that support stop-in-wait functionality have their clocks disabled in these configurations.

14.4.2.5 Transition from stop modes to Debug mode

The debugger module supports a transition from STOP, WAIT, VLPS, and VLPW back to a Halted state when the debugger has been enabled. As part of this transition, system clocking is re-established and is equivalent to the normal RUN and VLPR mode clocking configuration.

14.4.3 Run modes

The run modes supported by this device can be found here.

- Run (RUN)
- Very Low-Power Run (VLPR)

14.4.3.1 RUN mode

This is the normal operating mode for the device.

This mode is selected after any reset. When the ARM processor exits reset, it sets up the stack, program counter (PC), and link register (LR):

- The processor reads the start SP (SP_main) from vector-table offset 0x000
- The processor reads the start PC from vector-table offset 0x004
- LR is set to 0xFFFF_FFFF.

To reduce power in this mode, disable the clocks to unused modules.

14.4.3.2 Very-Low Power Run (VLPR) mode

In VLPR mode, the on-chip voltage regulator is put into a stop mode regulation state. In this state, the regulator is designed to supply enough current to the MCU over a reduced frequency. To further reduce power in this mode, disable the clocks to unused modules using their corresponding clock gating control bits in the SIM's registers.

Before entering this mode, the following conditions must be met:

- The MCG must be configured in a mode which is supported during VLPR. See the Power Management details for information about these MCG modes.
- All clock monitors in the MCG must be disabled.
- The maximum frequencies of the system, bus, flash, and core are restricted. See the Power Management details about which frequencies are supported.
- Mode protection must be set to allow VLP modes, that is, PMPROT[AVLP] is 1.
- PMCTRL[RUNM] must be set to 10b to enter VLPR.
- Flash programming/erasing is not allowed.

NOTE

Do not increase the clock frequency while in VLPR mode, because the regulator is slow in responding and cannot manage fast load transitions. In addition, do not modify the clock source in the MCG module or any clock divider registers. Module clock enables in the SIM can be set, but not cleared.

To reenter Normal Run mode, clear PMCTRL[RUNM]. PMSTAT is a read-only status register that can be used to determine when the system has completed an exit to RUN mode. When PMSTAT=RUN, the system is in run regulation and the MCU can run at full speed in any clock mode. If a higher execution frequency is desired, poll PMSTAT until it is set to RUN when returning from VLPR mode.

Any reset always causes an exit from VLPR and returns the device to RUN mode after the MCU exits its reset flow.

14.4.4 Wait modes

This device contains two different wait modes which are listed here.

- Wait
- Very-Low Power Wait (VLPW)

14.4.4.1 WAIT mode

WAIT mode is entered when the ARM core enters the Sleep-Now or Sleep-On-Exit modes while SLEEPDEEP is cleared. The ARM CPU enters a low-power state in which it is not clocked, but peripherals continue to be clocked provided they are enabled.

When an interrupt request occurs, the CPU exits WAIT mode and resumes processing in RUN mode, beginning with the stacking operations leading to the interrupt service routine.

A system reset causes an exit from WAIT mode, returning the device to normal RUN mode.

14.4.4.2 Very-Low-Power Wait (VLPW) mode

VLPW mode is entered by entering the Sleep-Now or Sleep-On-Exit mode while SLEEPDEEP is cleared and the device is in VLPR mode.

In VLPW, the on-chip voltage regulator remains in its stop regulation state. In this state, the regulator is designed to supply enough current to the device at a reduced frequency. To further reduce power in this mode, disable the clocks to unused modules.

VLPR mode restrictions also apply to VLPW.

When an interrupt from VLPW occurs, the device returns to VLPR mode to execute the interrupt service routine.

A system reset causes an exit from VLPW mode, returning the device to normal RUN mode.

14.4.5 Stop modes

This device contains a variety of stop modes to meet your application needs.

The stop modes range from:

- a stopped CPU, with all I/O, logic, and memory states retained, and certain asynchronous mode peripherals operating

to:

- a powered down CPU, with only I/O and a small register file retained, very few asynchronous mode peripherals operating, while the remainder of the MCU is powered down.

The choice of stop mode depends upon the user's application, and how power usage and state retention versus functional needs and recovery time may be traded off.

The various stop modes are selected by setting the appropriate fields in PMPROT and PMCTRL. The selected stop mode is entered during the sleep-now or sleep-on-exit entry with the SLEEPDEEP bit set in the System Control Register in the ARM core.

The available stop modes are:

- Normal Stop (STOP)
- Very-Low Power Stop (VLPS)
- Very-Low-Leakage Stop (VLLSx)

14.4.5.1 STOP mode

STOP mode is entered via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System Control Register in the ARM core.

The MCG module can be configured to leave the reference clocks running.

A module capable of providing an asynchronous interrupt to the device takes the device out of STOP mode and returns the device to normal RUN mode. Refer to the device's Power Management chapter for peripheral, I/O, and memory operation in STOP mode. When an interrupt request occurs, the CPU exits STOP mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

A system reset will cause an exit from STOP mode, returning the device to normal RUN mode via an MCU reset.

14.4.5.2 Very-Low-Power Stop (VLPS) mode

The two ways in which VLPS mode can be entered are listed here.

- Entry into stop via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System Control Register in the ARM core while the MCU is in VLPR mode and PMCTRL[STOPM] = 010 or 000.
- Entry into stop via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System Control Register in the ARM core while the MCU is in normal RUN mode and PMCTRL[STOPM] = 010. When VLPS is entered directly from RUN mode, exit to VLPR is disabled by hardware and the system will always exit back to RUN.

In VLPS, the on-chip voltage regulator remains in its stop regulation state as in VLPR.

A module capable of providing an asynchronous interrupt to the device takes the device out of VLPS and returns the device to VLPR mode.

A system reset will also cause a VLPS exit, returning the device to normal RUN mode.

14.4.5.3 Very-Low-Leakage Stop (VLLSx) modes

This device contains these very low leakage modes:

- VLLS3
- VLLS1
- VLLS0

VLLSx is often used in this document to refer to all of these modes.

All VLLSx modes can be entered from normal RUN or VLPR modes.

The MCU enters the configured VLLS mode if:

- In Sleep-Now or Sleep-On-Exit mode, the SLEEPDEEP bit is set in the System Control Register in the ARM core, and
- The device is configured as shown in Table 14-2.

In VLLS, the on-chip voltage regulator is in its stop-regulation state while most digital logic is powered off.

Before entering VLLS mode, the user should configure the Low-Leakage Wake-up (LLWU) module to enable the desired wakeup sources. The available wake-up sources in VLLS are detailed in the chip configuration details for this device.

After wakeup from VLLS, the device returns to normal RUN mode with a pending LLWU interrupt. In the LLWU interrupt service routine (ISR), the user can poll the LLWU module wake-up flags to determine the source of the wake-up.

When entering VLLS, each I/O pin is latched as configured before executing VLLS. Because all digital logic in the MCU is powered off, all port and peripheral data is lost during VLLS. This information must be restored before PMC_REGSC[ACKISO] is set.

An asserted $\overline{\text{RESET}}$ pin will cause an exit from any VLLS mode, returning the device to normal RUN mode. When exiting VLLS via the $\overline{\text{RESET}}$ pin, RCM_SRS[PIN] and RCM_SRS[WAKEUP] are set.

14.4.6 Debug in low power modes

When the MCU is secure, the device disables/limits debugger operation. When the MCU is unsecure, the ARM debugger can assert two power-up request signals:

- System power up, via SYSPWR in the Debug Port Control/Stat register
- Debug power up, via CDBGPWRUPREQ in the Debug Port Control/Stat register

When asserted while in RUN, WAIT, VLPR, or VLPW the mode controller drives a corresponding acknowledge for each signal, that is, both CDBGPWRUPACK and CSYSPWRUPACK. When both requests are asserted, the mode controller handles attempts to enter STOP and VLPS by entering an emulated stop state. In this emulated stop state:

- the regulator is in run regulation,
- the MCG-generated clock source is enabled,
- all system clocks, except the core clock, are disabled,
- the debug module has access to core registers, and
- access to the on-chip peripherals is blocked.

No debug is available while the MCU is in VLLS modes.

Entering into a VLLS mode causes all of the debug controls and settings to be powered off. To give time to the debugger to sync with the MCU, the MDM AP Control Register includes a Very-Low-Leakage Debug Request (VLLDBGREQ) bit that is set to configure the Reset Controller logic to hold the system in reset after the next recovery from a VLLS mode. This bit allows the debugger time to reinitialize the debug module before the debug session continues.

The MDM AP Control Register also includes a Very Low Leakage Debug Acknowledge (VLLDBGACK) bit that is set to release the ARM core being held in reset following a VLLS recovery. The debugger reinitializes all debug IP, and then asserts the VLLDBGACK control bit to allow the RCM to release the ARM core from reset and allow CPU operation to begin.

The VLLDBGACK bit is cleared by the debugger (or can be left set as is) or clears automatically due to the reset generated as part of the next VLLS recovery.

Chapter 15
Power Management Controller (PMC)

15.1 Introduction

NOTE

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

The power management controller (PMC) contains the internal voltage regulator, power on reset (POR), and low voltage detect system (LVD) for the VDD domain.

See AN4503: Power Management for Kinetis MCUs for further details on using the PMC.

15.2 Features

A list of included PMC features can be found here.

- Internal voltage regulator
- Active POR providing brown-out detect
- Low-voltage detect (LVD) on VDD supporting two low-voltage trip points with four warning levels per trip point

15.3 Low-voltage detect (LVD) system

This device includes a system to guard against low-voltage conditions on the VDD supply. When the VDD supply falls below a specific trip point, the LVD circuit puts the device into a reset state, preventing the device from attempting to operate below its operating voltage range.

The system is comprised of a power-on reset (POR) circuit and a LVD circuit with a user-selectable trip voltage: high (V_{LVDH}) or low (V_{LVDL}). The trip voltage is selected by LVDSC1[LVDV]. The LVD is disabled upon entering VLPx and VLLSx modes.

Two flags are available to indicate the status of the low-voltage detect system:

- The Low Voltage Detect Flag in the Low Voltage Status and Control 1 Register (LVDSC1[LVDF]) operates in a level sensitive manner. LVDSC1[LVDF] is set when the supply voltage falls below the selected trip point (VLVD). LVDSC1[LVDF] is cleared by writing 1 to LVDSC1[LVDACK], but only if the internal supply has returned above the trip point; otherwise, LVDSC1[LVDF] remains set.
- The Low Voltage Warning Flag (LVWF) in the Low Voltage Status and Control 2 Register (LVDSC2[LVWF]) operates in a level sensitive manner. LVDSC2[LVWF] is set when the supply voltage falls below the selected monitor trip point (VLVW). LVDSC2[LVWF] is cleared by writing one to LVDSC2[LVWACK], but only if the internal supply has returned above the trip point; otherwise, LVDSC2[LVWF] remains set.

15.3.1 LVD reset operation

By setting LVDSC1[LVDRE], the LVD generates a reset upon detection of a low-voltage condition. The low-voltage detection threshold is determined by LVDSC1[LVDV]. After an LVD reset occurs, the LVD system holds the MCU in reset until the supply voltage rises above this threshold. The LVD field in the SRS register of the RCM module (RCM_SRS[LVD]) is set following an LVD or power-on reset.

15.3.2 LVD interrupt operation

By configuring the LVD circuit for interrupt operation (LVDSC1[LVDIE] set and LVDSC1[LVDRE] clear), LVDSC1[LVDF] is set and an LVD interrupt request occurs upon detection of a low voltage condition. LVDSC1[LVDF] is cleared by writing 1 to LVDSC1[LVDACK].

15.3.3 Low-voltage warning (LVW) interrupt operation

The LVD system that monitors the VDD supply contains a Low-Voltage Warning Flag (LVWF) in the Low Voltage Detect Status and Control 2 Register to indicate that the supply voltage is approaching, but is above, the LVD voltage. The LVW also has an

interrupt, which is enabled by setting LVDSC2[LVWIE]. If enabled, an LVW interrupt request occurs when LVDSC2[LVWF] is set. LVDSC2[LVWF] is cleared by writing 1 to LVDSC2[LVWACK].

LVDSC2[LVWV] selects one of the four trip voltages:

- Highest: V_{LVW4}
- Two mid-levels: V_{LVW3} and V_{LVW2}
- Lowest: V_{LVW1}

15.4 I/O retention

When in VLLS modes, the I/O states are held on a wake-up event (with the exception of wake-up by reset event) until the wake-up has been acknowledged via a write to REGSC[ACKISO]. In the case of VLLS exit via a RESET pin, the I/O are released and default to their reset state. In this case, no write to REGSC[ACKISO] is needed.

15.5 Memory map and register descriptions

Details about the PMC registers can be found here.

NOTE

Different portions of PMC registers are reset only by particular reset types. Each register's description provides details. For more information about the types of reset on this chip, refer to the Reset section details.

The PMC registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

PMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_D000	Low Voltage Detect Status And Control 1 register (PMC_LVDSC1)	8	R/W	10h	15.5.1/310
4007_D001	Low Voltage Detect Status And Control 2 register (PMC_LVDSC2)	8	R/W	00h	15.5.2/311
4007_D002	Regulator Status And Control register (PMC_REGSC)	8	R/W	04h	15.5.3/312

15.5.1 Low Voltage Detect Status And Control 1 register (PMC_LVDSC1)

This register contains status and control bits to support the low voltage detect function. This register should be written during the reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

While the device is in the very low power or low leakage modes, the LVD system is disabled regardless of LVDSC1 settings. To protect systems that must have LVD always on, configure the Power Mode Protection (PMPROT) register of the SMC module (SMC_PMPROT) to disallow any very low power or low leakage modes from being enabled.

See the device's data sheet for the exact LVD trip voltages.

NOTE

The LVDV bits are reset solely on a POR Only event. The register's other bits are reset on Chip Reset Not VLLS. For more information about these reset types, refer to the Reset section details.

Address: 4007_D000h base + 0h offset = 4007_D000h

Bit	7	6	5	4	3	2	1	0
Read	LVDF	0	LVDIE	LVDRE	0		LVDV	
Write		LVDACK						
Reset	0	0	0	1	0	0	0	0

PMC_LVDSC1 field descriptions

Field	Description
7 LVDF	Low-Voltage Detect Flag This read-only status field indicates a low-voltage detect event. 0 Low-voltage event not detected 1 Low-voltage event detected
6 LVDACK	Low-Voltage Detect Acknowledge This write-only field is used to acknowledge low voltage detection errors. Write 1 to clear LVDF. Reads always return 0.
5 LVDIE	Low-Voltage Detect Interrupt Enable Enables hardware interrupt requests for LVDF. 0 Hardware interrupt disabled (use polling) 1 Request a hardware interrupt when LVDF = 1

Table continues on the next page...

PMC_LVDSC1 field descriptions (continued)

Field	Description
4 LVDRE	Low-Voltage Detect Reset Enable This write-once bit enables LVDF events to generate a hardware reset. Additional writes are ignored. 0 LVDF does not generate hardware resets 1 Force an MCU reset when LVDF = 1
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
LVDV	Low-Voltage Detect Voltage Select Selects the LVD trip point voltage (V_{LVD}). 00 Low trip point selected ($V_{LVD} = V_{LVDL}$) 01 High trip point selected ($V_{LVD} = V_{LVDH}$) 10 Reserved 11 Reserved

15.5.2 Low Voltage Detect Status And Control 2 register (PMC_LVDSC2)

This register contains status and control bits to support the low voltage warning function.

While the device is in the very low power or low leakage modes, the LVD system is disabled regardless of LVDSC2 settings.

See the device's data sheet for the exact LVD trip voltages.

NOTE

The LVW trip voltages depend on LVWV and LVDV.

NOTE

LVWV is reset solely on a POR Only event. The other fields of the register are reset on Chip Reset Not VLLS. For more information about these reset types, refer to the Reset section details.

Address: 4007_D000h base + 1h offset = 4007_D001h

Bit	7	6	5	4	3	2	1	0
Read	LVWF	0	LVWIE	0			LVWV	
Write		LVWACK						
Reset	0	0	0	0	0	0	0	0

PMC_LVDSC2 field descriptions

Field	Description
7 LVWF	Low-Voltage Warning Flag This read-only status field indicates a low-voltage warning event. LVWF is set when V_{Supply} transitions below the trip point, or after reset and V_{Supply} is already below V_{LVW}. LVWF may be 1 after power-on reset, therefore, to use LVW interrupt function, before enabling LVWIE, LVWF must be cleared by writing LVWACK first. 0 Low-voltage warning event not detected 1 Low-voltage warning event detected
6 LVWACK	Low-Voltage Warning Acknowledge This write-only field is used to acknowledge low voltage warning errors. Write 1 to clear LVWF. Reads always return 0.
5 LVWIE	Low-Voltage Warning Interrupt Enable Enables hardware interrupt requests for LVWF. 0 Hardware interrupt disabled (use polling) 1 Request a hardware interrupt when LVWF = 1
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
LVWV	Low-Voltage Warning Voltage Select Selects the LVW trip point voltage (V_{LVW}). The actual voltage for the warning depends on LVDSC1[LVDV]. 00 Low trip point selected ($V_{LVW} = V_{LVW1}$) 01 Mid 1 trip point selected ($V_{LVW} = V_{LVW2}$) 10 Mid 2 trip point selected ($V_{LVW} = V_{LVW3}$) 11 High trip point selected ($V_{LVW} = V_{LVW4}$)

15.5.3 Regulator Status And Control register (PMC_REGSC)

The PMC contains an internal voltage regulator. The voltage regulator design uses a bandgap reference that is also available through a buffer as input to certain internal peripherals, such as the CMP and ADC. The internal regulator provides a status bit (REGONS) indicating the regulator is in run regulation.

NOTE

This register is reset on Chip Reset Not VLLS and by reset types that trigger Chip Reset not VLLS. See the Reset section details for more information.

Address: 4007_D000h base + 2h offset = 4007_D002h

Bit	7	6	5	4	3	2	1	0
Read	0	0	Reserved	BGEN	ACKISO	REGONS	Reserved	BGBE
Write					w1c			
Reset	0	0	0	0	0	1	0	0

PMC_REGSC field descriptions

Field	Description
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 read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved.
4 BGEN	Bandgap Enable In VLPx Operation BGEN controls whether the bandgap is enabled in lower power modes of operation (VLPx, and VLLSx). When on-chip peripherals require the bandgap voltage reference in low power modes of operation, set BGEN to continue to enable the bandgap operation. **NOTE:** When the bandgap voltage reference is not needed in low power modes, clear BGEN to avoid excess power consumption. 0 Bandgap voltage reference is disabled in VLPx , and VLLSx modes. 1 Bandgap voltage reference is enabled in VLPx , and VLLSx modes.
3 ACKISO	Acknowledge Isolation Reading this field indicates whether certain peripherals and the I/O pads are in a latched state as a result of having been in a VLLS mode. Writing 1 to this field when it is set releases the I/O pads and certain peripherals to their normal run mode state. **NOTE:** After recovering from a VLLS mode, user should restore chip configuration before clearing ACKISO. In particular, pin configuration for enabled LLWU wakeup pins should be restored to avoid any LLWU flag from being falsely set when ACKISO is cleared. 0 Peripherals and I/O pads are in normal run state. 1 Certain peripherals and I/O pads are in an isolated and latched state.
2 REGONS	Regulator In Run Regulation Status This read-only field provides the current status of the internal voltage regulator. 0 Regulator is in stop regulation or in transition to/from it 1 Regulator is in run regulation
1 Reserved	This field is reserved. **NOTE:** This reserved bit must remain cleared (set to 0).
0 BGBE	Bandgap Buffer Enable Enables the bandgap buffer. 0 Bandgap buffer not enabled 1 Bandgap buffer enabled

Chapter 16
Low-Leakage Wakeup Unit (LLWU)

16.1 Introduction

NOTE

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

The LLWU module allows the user to select up to 32 external pins and up to 8 internal modules as interrupt wake-up sources from low-leakage power modes. It also supports up to 8 internal modules as temporary DMA wake-up sources.

The input sources are described in the device's chip configuration details. Each of the available wake-up sources can be individually enabled.

The $\overline{\text{RESET}}$ pin is an additional source for triggering an exit from low-leakage power modes, and causes the MCU to exit VLLS through a reset flow.

The LLWU module also includes four optional digital pin filters for the external wakeup pins.

See AN4503: Power Management for Kinetis MCUs for further details on using the LLWU.

16.1.1 Features

The LLWU module features include:

- Support for up to 32 external input pins and up to 8 internal modules with individual enable bits for MCU interrupt from low leakage modes
- Support for up to 8 internal modules with individual enable bits for DMA wakeup from low leakage modes

- Input sources may be external pins or from internal peripherals capable of running in VLLS. See the chip configuration information for wakeup input sources for this device.
- External pin wake-up inputs, each of which is programmable as falling-edge, rising-edge, or any change
- Wake-up inputs that are activated after MCU enters a low-leakage power mode
- Optional digital filters provided to qualify an external pin detect. Note that when the LPO clock is disabled, the filters are disabled and bypassed.

16.1.2 Modes of operation

The LLWU module becomes functional on entry into a low-leakage power mode. After recovery from VLLS, the LLWU continues to detect wake-up events until the user has acknowledged the wake-up via a write to PMC_REGSC[ACKISO].

16.1.2.1 VLLS modes

All wakeup and reset events result in VLLS exit via a reset flow.

16.1.2.2 Non-low leakage modes

The LLWU is not active in all non-low leakage modes where detection and control logic are in a static state. The LLWU registers are accessible in non-low leakage modes and are available for configuring and reading status when bus transactions are possible.

When the wake-up pin filters are enabled, filter operation begins immediately. If a low leakage mode is entered within five LPO clock cycles of an active edge, the edge event will be detected by the LLWU.

16.1.2.3 Debug mode

When the chip is in Debug mode and then enters a VLLSx mode, no debug logic works in the fully-functional low-leakage mode. Upon an exit from the VLLSx mode, the LLWU becomes inactive.

16.1.3 Block diagram

The following figure is the block diagram for the LLWU module.

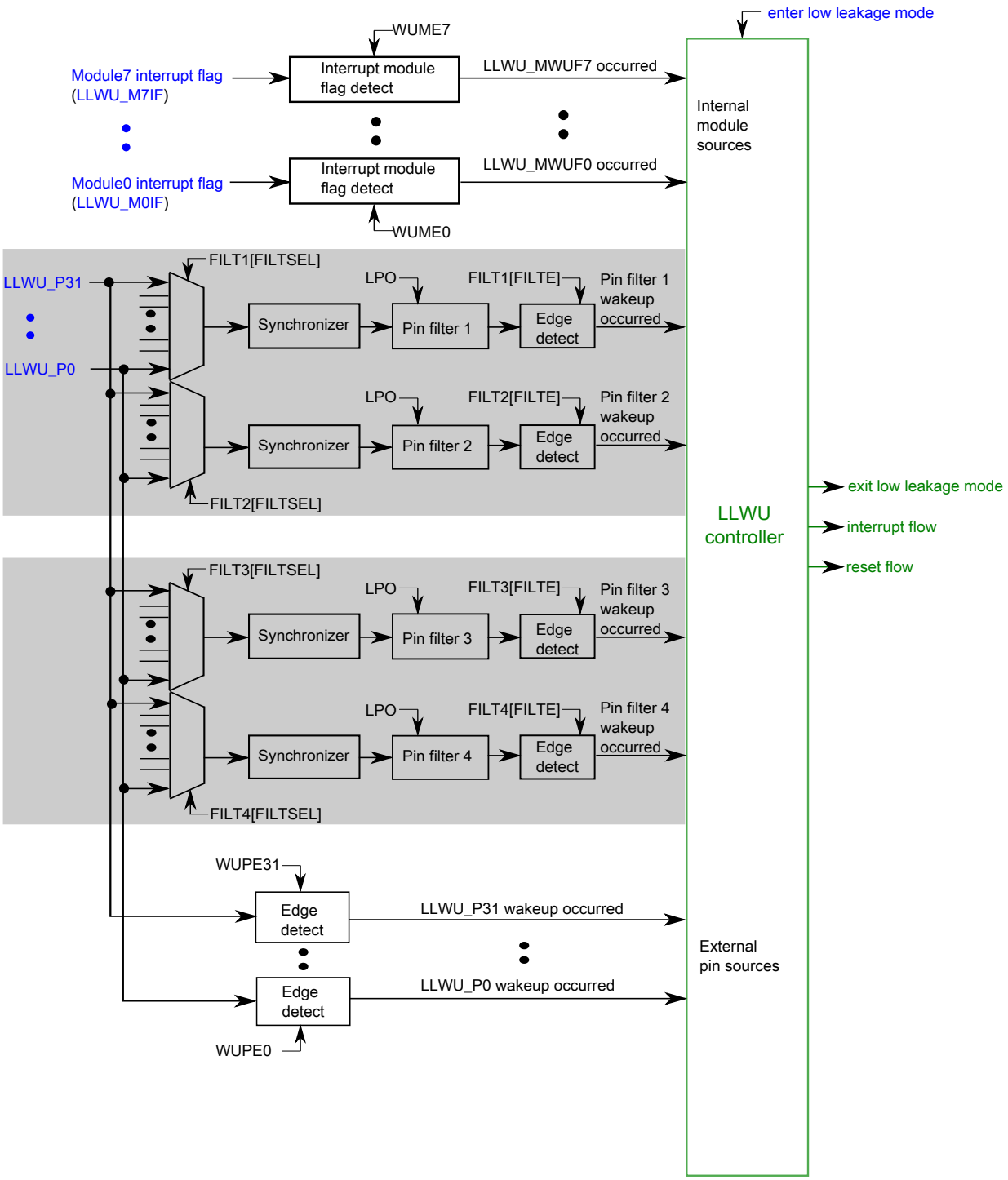


Figure 16-1. LLWU block diagram

16.2 LLWU signal descriptions

The signal properties of LLWU are shown in the table found here.

The external wakeup input pins can be enabled to detect either rising-edge, falling-edge, or on any change.

Table 16-1. LLWU signal descriptions

Signal	Description	I/O
LLWU_Pn	Wakeup inputs (n = 0- 31)	I

16.3 Memory map/register definition

The LLWU includes the following registers:

- Wake-up source enable registers
  - Enable external pin input sources
  - Enable internal peripheral interrupt sources
  - Enable internal peripheral DMA sources
- Wake-up flag registers
  - Indication of wakeup source that caused exit from a low-leakage power mode includes external pin or internal module interrupt
- Wake-up pin filter enable registers

NOTE

The LLWU registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

All LLWU registers are reset by Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. Each register's displayed reset value represents this subset of reset types. LLWU registers are unaffected by reset types that do not trigger Chip Reset not VLLS. For more information about the types of reset on this chip, refer to the Introduction details.

LLWU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_C000	LLWU Pin Enable 1 register (LLWU_PE1)	8	R/W	00h	16.3.1/319
4007_C001	LLWU Pin Enable 2 register (LLWU_PE2)	8	R/W	00h	16.3.2/320
4007_C002	LLWU Pin Enable 3 register (LLWU_PE3)	8	R/W	00h	16.3.3/321
4007_C003	LLWU Pin Enable 4 register (LLWU_PE4)	8	R/W	00h	16.3.4/322
4007_C004	LLWU Pin Enable 5 register (LLWU_PE5)	8	R/W	00h	16.3.5/323
4007_C005	LLWU Pin Enable 6 register (LLWU_PE6)	8	R/W	00h	16.3.6/325
4007_C006	LLWU Pin Enable 7 register (LLWU_PE7)	8	R/W	00h	16.3.7/326
4007_C007	LLWU Pin Enable 8 register (LLWU_PE8)	8	R/W	00h	16.3.8/327
4007_C008	LLWU Module Enable register (LLWU_ME)	8	R/W	00h	16.3.9/328
4007_C009	LLWU Pin Flag 1 register (LLWU_PF1)	8	R/W	00h	16.3.10/329
4007_C00A	LLWU Pin Flag 2 register (LLWU_PF2)	8	R/W	00h	16.3.11/331
4007_C00B	LLWU Pin Flag 3 register (LLWU_PF3)	8	R/W	00h	16.3.12/333
4007_C00C	LLWU Pin Flag 4 register (LLWU_PF4)	8	R/W	00h	16.3.13/334
4007_C00D	LLWU Module Flag 5 register (LLWU_MF5)	8	R	00h	16.3.14/336
4007_C00E	LLWU Pin Filter 1 register (LLWU_FILT1)	8	R/W	00h	16.3.15/338
4007_C00F	LLWU Pin Filter 2 register (LLWU_FILT2)	8	R/W	00h	16.3.16/339

16.3.1 LLWU Pin Enable 1 register (LLWU_PE1)

LLWU_PE1 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P3-LLWU_P0.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 0h offset = 4007_C000h

Bit	7	6	5	4	3	2	1	0
Read Write	WUPE3		WUPE2		WUPE1		WUPE0	
Reset	0	0	0	0	0	0	0	0

LLWU_PE1 field descriptions

Field	Description
7–6 WUPE3	Wakeup Pin Enable For LLWU_P3 Enables and configures the edge detection for the wakeup pin.

Table continues on the next page...

LLWU_PE1 field descriptions (continued)

Field	Description
	00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE2	Wakeup Pin Enable For LLWU_P2 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE1	Wakeup Pin Enable For LLWU_P1 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE0	Wakeup Pin Enable For LLWU_P0 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

16.3.2 LLWU Pin Enable 2 register (LLWU_PE2)

LLWU_PE2 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P7-LLWU_P4.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 1h offset = 4007_C001h

Bit	7	6	5	4	3	2	1	0
Read Write	WUPE7		WUPE6		WUPE5		WUPE4	
Reset	0	0	0	0	0	0	0	0

LLWU_PE2 field descriptions

Field	Description
7–6 WUPE7	Wakeup Pin Enable For LLWU_P7 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE6	Wakeup Pin Enable For LLWU_P6 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE5	Wakeup Pin Enable For LLWU_P5 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE4	Wakeup Pin Enable For LLWU_P4 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

16.3.3 LLWU Pin Enable 3 register (LLWU_PE3)

LLWU_PE3 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P11-LLWU_P8.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 2h offset = 4007_C002h

Bit	7	6	5	4	3	2	1	0
Read Write	WUPE11		WUPE10		WUPE9		WUPE8	
Reset	0	0	0	0	0	0	0	0

LLWU_PE3 field descriptions

Field	Description
7–6 WUPE11	Wakeup Pin Enable For LLWU_P11 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE10	Wakeup Pin Enable For LLWU_P10 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE9	Wakeup Pin Enable For LLWU_P9 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE8	Wakeup Pin Enable For LLWU_P8 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

16.3.4 LLWU Pin Enable 4 register (LLWU_PE4)

LLWU_PE4 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P15-LLWU_P12.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset

types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 3h offset = 4007_C003h

Bit	7	6	5	4	3	2	1	0
Read Write	WUPE15		WUPE14		WUPE13		WUPE12	
Reset	0	0	0	0	0	0	0	0

LLWU_PE4 field descriptions

Field	Description
7–6 WUPE15	Wakeup Pin Enable For LLWU_P15 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE14	Wakeup Pin Enable For LLWU_P14 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE13	Wakeup Pin Enable For LLWU_P13 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE12	Wakeup Pin Enable For LLWU_P12 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

16.3.5 LLWU Pin Enable 5 register (LLWU_PE5)

LLWU_PE5 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P19-LLWU_P16.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 4h offset = 4007_C004h

Bit	7	6	5	4	3	2	1	0
Read Write	WUPE19		WUPE18		WUPE17		WUPE16	
Reset	0	0	0	0	0	0	0	0

LLWU_PE5 field descriptions

Field	Description
7–6 WUPE19	Wakeup Pin Enable For LLWU_P19 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE18	Wakeup Pin Enable For LLWU_P18 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE17	Wakeup Pin Enable For LLWU_P17 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE16	Wakeup Pin Enable For LLWU_P16 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

16.3.6 LLWU Pin Enable 6 register (LLWU_PE6)

LLWU_PE6 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P23-LLWU_P20.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 5h offset = 4007_C005h

Bit	7	6	5	4	3	2	1	0
Read Write	WUPE23		WUPE22		WUPE21		WUPE20	
Reset	0	0	0	0	0	0	0	0

LLWU_PE6 field descriptions

Field	Description
7–6 WUPE23	Wakeup Pin Enable For LLWU_P23 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE22	Wakeup Pin Enable For LLWU_P22 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE21	Wakeup Pin Enable For LLWU_P21 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE20	Wakeup Pin Enable For LLWU_P20 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection

Table continues on the next page...

LLWU_PE6 field descriptions (continued)

Field	Description
	10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

16.3.7 LLWU Pin Enable 7 register (LLWU_PE7)

LLWU_PE7 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P27-LLWU_P24.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 6h offset = 4007_C006h

Bit	7	6	5	4	3	2	1	0
Read Write	WUPE27		WUPE26		WUPE25		WUPE24	
Reset	0	0	0	0	0	0	0	0

LLWU_PE7 field descriptions

Field	Description
7–6 WUPE27	Wakeup Pin Enable For LLWU_P27 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE26	Wakeup Pin Enable For LLWU_P26 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE25	Wakeup Pin Enable For LLWU_P25 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection

Table continues on the next page...

LLWU_PE7 field descriptions (continued)

Field	Description
	10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE24	Wakeup Pin Enable For LLWU_P24 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

16.3.8 LLWU Pin Enable 8 register (LLWU_PE8)

LLWU_PE8 contains the field to enable and select the edge detect type for the external wakeup input pins LLWU_P31-LLWU_P28.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 7h offset = 4007_C007h

Bit	7	6	5	4	3	2	1	0
Read Write	WUPE31		WUPE30		WUPE29		WUPE28	
Reset	0	0	0	0	0	0	0	0

LLWU_PE8 field descriptions

Field	Description
7–6 WUPE31	Wakeup Pin Enable For LLWU_P31 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
5–4 WUPE30	Wakeup Pin Enable For LLWU_P30 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection

Table continues on the next page...

LLWU_PE8 field descriptions (continued)

Field	Description
	10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
3–2 WUPE29	Wakeup Pin Enable For LLWU_P29 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection
WUPE28	Wakeup Pin Enable For LLWU_P28 Enables and configures the edge detection for the wakeup pin. 00 External input pin disabled as wakeup input 01 External input pin enabled with rising edge detection 10 External input pin enabled with falling edge detection 11 External input pin enabled with any change detection

16.3.9 LLWU Module Enable register (LLWU_ME)

LLWU_ME contains the bits to enable the internal module flag as a wakeup input source for inputs MWUF7-MWUF0.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 8h offset = 4007_C008h

Bit	7	6	5	4	3	2	1	0
Read Write	WUME7	WUME6	WUME5	WUME4	WUME3	WUME2	WUME1	WUME0
Reset	0	0	0	0	0	0	0	0

LLWU_ME field descriptions

Field	Description
7 WUME7	Wakeup Module Enable For Module 7 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source

Table continues on the next page...

LLWU_ME field descriptions (continued)

Field	Description
6 WUME6	Wakeup Module Enable For Module 6 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
5 WUME5	Wakeup Module Enable For Module 5 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
4 WUME4	Wakeup Module Enable For Module 4 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
3 WUME3	Wakeup Module Enable For Module 3 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
2 WUME2	Wakeup Module Enable For Module 2 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
1 WUME1	Wakeup Module Enable for Module 1 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source
0 WUME0	Wakeup Module Enable For Module 0 Enables an internal module as a wakeup source input. 0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source

16.3.10 LLWU Pin Flag 1 register (LLWU_PF1)

LLWU_PF1 contains the wakeup flags indicating which wakeup source caused the MCU to exit VLLS mode. For VLLS, this is the source causing the MCU reset flow.

The external wakeup flags are read-only and clearing a flag is accomplished by a write of a 1 to the corresponding WUFx bit. The wakeup flag (WUFx), if set, will remain set if the associated WUPEx bit is cleared.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + 9h offset = 4007_C009h

Bit	7	6	5	4	3	2	1	0
Read	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1	WUF0
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

LLWU_PF1 field descriptions

Field	Description
7 WUF7	Wakeup Flag For LLWU_P7 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF7. 0 LLWU_P7 input was not a wakeup source 1 LLWU_P7 input was a wakeup source
6 WUF6	Wakeup Flag For LLWU_P6 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF6. 0 LLWU_P6 input was not a wakeup source 1 LLWU_P6 input was a wakeup source
5 WUF5	Wakeup Flag For LLWU_P5 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF5. 0 LLWU_P5 input was not a wakeup source 1 LLWU_P5 input was a wakeup source
4 WUF4	Wakeup Flag For LLWU_P4 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF4. 0 LLWU_P4 input was not a wakeup source 1 LLWU_P4 input was a wakeup source
3 WUF3	Wakeup Flag For LLWU_P3 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF3.

Table continues on the next page...

LLWU_PF1 field descriptions (continued)

Field	Description
	0 LLWU_P3 input was not a wakeup source 1 LLWU_P3 input was a wakeup source
2 WUF2	Wakeup Flag For LLWU_P2 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF2. 0 LLWU_P2 input was not a wakeup source 1 LLWU_P2 input was a wakeup source
1 WUF1	Wakeup Flag For LLWU_P1 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF1. 0 LLWU_P1 input was not a wakeup source 1 LLWU_P1 input was a wakeup source
0 WUF0	Wakeup Flag For LLWU_P0 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF0. 0 LLWU_P0 input was not a wakeup source 1 LLWU_P0 input was a wakeup source

16.3.11 LLWU Pin Flag 2 register (LLWU_PF2)

LLWU_PF2 contains the wakeup flags indicating which wakeup source caused the MCU to exit or VLLS mode. For VLLS, this is the source causing the MCU reset flow.

The external wakeup flags are read-only and clearing a flag is accomplished by a write of a 1 to the corresponding WUFx bit. The wakeup flag (WUFx), if set, will remain set if the associated WUPEx bit is cleared.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + Ah offset = 4007_C00Ah

Bit	7	6	5	4	3	2	1	0
Read	WUF15	WUF14	WUF13	WUF12	WUF11	WUF10	WUF9	WUF8
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

LLWU_PF2 field descriptions

Field	Description
7 WUF15	Wakeup Flag For LLWU_P15 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF15. 0 LLWU_P15 input was not a wakeup source 1 LLWU_P15 input was a wakeup source
6 WUF14	Wakeup Flag For LLWU_P14 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF14. 0 LLWU_P14 input was not a wakeup source 1 LLWU_P14 input was a wakeup source
5 WUF13	Wakeup Flag For LLWU_P13 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF13. 0 LLWU_P13 input was not a wakeup source 1 LLWU_P13 input was a wakeup source
4 WUF12	Wakeup Flag For LLWU_P12 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF12. 0 LLWU_P12 input was not a wakeup source 1 LLWU_P12 input was a wakeup source
3 WUF11	Wakeup Flag For LLWU_P11 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF11. 0 LLWU_P11 input was not a wakeup source 1 LLWU_P11 input was a wakeup source
2 WUF10	Wakeup Flag For LLWU_P10 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF10. 0 LLWU_P10 input was not a wakeup source 1 LLWU_P10 input was a wakeup source
1 WUF9	Wakeup Flag For LLWU_P9 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF9. 0 LLWU_P9 input was not a wakeup source 1 LLWU_P9 input was a wakeup source
0 WUF8	Wakeup Flag For LLWU_P8

Table continues on the next page...

LLWU_PF2 field descriptions (continued)

Field	Description
	Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF8. 0 LLWU_P8 input was not a wakeup source 1 LLWU_P8 input was a wakeup source

16.3.12 LLWU Pin Flag 3 register (LLWU_PF3)

LLWU_PF3 contains the wakeup flags indicating which wakeup source caused the MCU to exit VLLS mode. For VLLS, this is the source causing the MCU reset flow.

The external wakeup flags are read-only and clearing a flag is accomplished by a write of a 1 to the corresponding WUFx bit. The wakeup flag (WUFx), if set, will remain set if the associated WUPEx bit is cleared.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + Bh offset = 4007_C00Bh

Bit	7	6	5	4	3	2	1	0
Read	WUF23	WUF22	WUF21	WUF20	WUF19	WUF18	WUF17	WUF16
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

LLWU_PF3 field descriptions

Field	Description
7 WUF23	Wakeup Flag For LLWU_P23 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF23. 0 LLWU_P23 input was not a wakeup source 1 LLWU_P23 input was a wakeup source
6 WUF22	Wakeup Flag For LLWU_P22 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF22. 0 LLWU_P22 input was not a wakeup source 1 LLWU_P22 input was a wakeup source

Table continues on the next page...

LLWU_PF3 field descriptions (continued)

Field	Description
5 WUF21	Wakeup Flag For LLWU_P21 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF21. 0 LLWU_P21 input was not a wakeup source 1 LLWU_P21 input was a wakeup source
4 WUF20	Wakeup Flag For LLWU_P20 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF20. 0 LLWU_P20 input was not a wakeup source 1 LLWU_P20 input was a wakeup source
3 WUF19	Wakeup Flag For LLWU_P19 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF19. 0 LLWU_P19 input was not a wakeup source 1 LLWU_P19 input was a wakeup source
2 WUF18	Wakeup Flag For LLWU_P18 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF18. 0 LLWU_P18 input was not a wakeup source 1 LLWU_P18 input was a wakeup source
1 WUF17	Wakeup Flag For LLWU_P17 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF17. 0 LLWU_P17 input was not a wakeup source 1 LLWU_P17 input was a wakeup source
0 WUF16	Wakeup Flag For LLWU_P16 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF16. 0 LLWU_P16 input was not a wakeup source 1 LLWU_P16 input was a wakeup source

16.3.13 LLWU Pin Flag 4 register (LLWU_PF4)

LLWU_PF4 contains the wakeup flags indicating which wakeup source caused the MCU to exit or VLLS mode. For VLLS, this is the source causing the MCU reset flow.

The external wakeup flags are read-only and clearing a flag is accomplished by a write of a 1 to the corresponding WUFx bit. The wakeup flag (WUFx), if set, will remain set if the associated WUPEx bit is cleared.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + Ch offset = 4007_C00Ch

Bit	7	6	5	4	3	2	1	0
Read	WUF31	WUF30	WUF29	WUF28	WUF27	WUF26	WUF25	WUF24
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

LLWU_PF4 field descriptions

Field	Description
7 WUF31	Wakeup Flag For LLWU_P31 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF31. 0 LLWU_P31 input was not a wakeup source 1 LLWU_P31 input was a wakeup source
6 WUF30	Wakeup Flag For LLWU_P30 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF30. 0 LLWU_P30 input was not a wakeup source 1 LLWU_P30 input was a wakeup source
5 WUF29	Wakeup Flag For LLWU_P29 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF29. 0 LLWU_P29 input was not a wakeup source 1 LLWU_P29 input was a wakeup source
4 WUF28	Wakeup Flag For LLWU_P28 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF28. 0 LLWU_P28 input was not a wakeup source 1 LLWU_P28 input was a wakeup source
3 WUF27	Wakeup Flag For LLWU_P27 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF27.

Table continues on the next page...

LLWU_PF4 field descriptions (continued)

Field	Description
	0 LLWU_P27 input was not a wakeup source 1 LLWU_P27 input was a wakeup source
2 WUF26	Wakeup Flag For LLWU_P26 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF26. 0 LLWU_P26 input was not a wakeup source 1 LLWU_P26 input was a wakeup source
1 WUF25	Wakeup Flag For LLWU_P25 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF25. 0 LLWU_P25 input was not a wakeup source 1 LLWU_P25 input was a wakeup source
0 WUF24	Wakeup Flag For LLWU_P24 Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag write a one to WUF24. 0 LLWU_P24 input was not a wakeup source 1 LLWU_P24 input was a wakeup source

16.3.14 LLWU Module Flag 5 register (LLWU_MF5)

LLWU_MF5 contains the wakeup flags indicating which internal wakeup source caused the MCU to exit VLLS mode. For VLLS, this is the source causing the MCU reset flow.

For internal peripherals that are capable of running in a low-leakage power mode, such as a real time clock module or CMP module, the flag from the associated peripheral is accessible as the MWUFx bit. The flag will need to be cleared in the peripheral instead of writing a 1 to the MWUFx bit.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + Dh offset = 4007_C00Dh

Bit	7	6	5	4	3	2	1	0
Read	MWUF7	MWUF6	MWUF5	MWUF4	MWUF3	MWUF2	MWUF1	MWUF0
Write								
Reset	0	0	0	0	0	0	0	0

LLWU_MF5 field descriptions

Field	Description
7 MWUF7	Wakeup flag For module 7 Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism. 0 Module 7 input was not a wakeup source 1 Module 7 input was a wakeup source
6 MWUF6	Wakeup flag For module 6 Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism. 0 Module 6 input was not a wakeup source 1 Module 6 input was a wakeup source
5 MWUF5	Wakeup flag For module 5 Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism. 0 Module 5 input was not a wakeup source 1 Module 5 input was a wakeup source
4 MWUF4	Wakeup flag For module 4 Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism. 0 Module 4 input was not a wakeup source 1 Module 4 input was a wakeup source
3 MWUF3	Wakeup flag For module 3 Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism. 0 Module 3 input was not a wakeup source 1 Module 3 input was a wakeup source
2 MWUF2	Wakeup flag For module 2 Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism. 0 Module 2 input was not a wakeup source 1 Module 2 input was a wakeup source
1 MWUF1	Wakeup flag For module 1 Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism. 0 Module 1 input was not a wakeup source 1 Module 1 input was a wakeup source
0 MWUF0	Wakeup flag For module 0

Table continues on the next page...

LLWU_MF5 field descriptions (continued)

Field	Description
	Indicates that an enabled internal peripheral was a source of exiting a low-leakage power mode. To clear the flag, follow the internal peripheral flag clearing mechanism. 0 Module 0 input was not a wakeup source 1 Module 0 input was a wakeup source

16.3.15 LLWU Pin Filter 1 register (LLWU_FILT1)

LLWU_FILT1 is a control and status register that is used to enable/disable the digital filter 1 features for an external pin.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + Eh offset = 4007_C00Eh

Bit	7	6	5	4	3	2	1	0
Read	FILTF	FILTE			FILTSEL			
Write	w1c							
Reset	0	0	0	0	0	0	0	0

LLWU_FILT1 field descriptions

Field	Description
7 FILTF	Filter Detect Flag Indicates that the filtered external wakeup pin, selected by FILTSEL, was a source of exiting a low-leakage power mode. To clear the flag write a one to FILTF. 0 Pin Filter 1 was not a wakeup source 1 Pin Filter 1 was a wakeup source
6–5 FILTE	Digital Filter On External Pin Controls the digital filter options for the external pin detect. 00 Filter disabled 01 Filter posedge detect enabled 10 Filter negedge detect enabled 11 Filter any edge detect enabled
FILTSEL	Filter Pin Select Selects 1 of the wakeup pins to be muxed into the filter.

Table continues on the next page...

LLWU_FILT1 field descriptions (continued)

Field	Description
	00000 Select LLWU_P0 for filter ... ... 11111 Select LLWU_P31 for filter

16.3.16 LLWU Pin Filter 2 register (LLWU_FILT2)

LLWU_FILT2 is a control and status register that is used to enable/disable the digital filter 2 features for an external pin.

NOTE

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Introduction details for more information.

Address: 4007_C000h base + Fh offset = 4007_C00Fh

Bit	7	6	5	4	3	2	1	0
Read	FILTF	FILTE			FILTSEL			
Write	w1c							
Reset	0	0	0	0	0	0	0	0

LLWU_FILT2 field descriptions

Field	Description
7 FILTF	Filter Detect Flag Indicates that the filtered external wakeup pin, selected by FILTSEL, was a source of exiting a low-leakage power mode. To clear the flag write a one to FILTF. 0 Pin Filter 2 was not a wakeup source 1 Pin Filter 2 was a wakeup source
6–5 FILTE	Digital Filter On External Pin Controls the digital filter options for the external pin detect. 00 Filter disabled 01 Filter posedge detect enabled 10 Filter negedge detect enabled 11 Filter any edge detect enabled
FILTSEL	Filter Pin Select Selects 1 of the wakeup pins to be muxed into the filter. 00000 Select LLWU_P0 for filter

Table continues on the next page...

LLWU_FILT2 field descriptions (continued)

Field	Description
	... ... 11111 Select LLWU_P31 for filter

16.4 Functional description

Thie low-leakage wakeup unit (LLWU) module allows internal peripherals and external input pins as a source of wakeup from low-leakage modes.

It is operational only in VLLSx modes.

The LLWU module contains pin enables for each external pin and internal module. For each external pin, the user can disable or select the edge type for the wakeup with the following options:

- Falling-edge
- Rising-edge
- Either-edge

When an external pin is enabled as a wakeup source, the pin must be configured as an input pin.

The LLWU implements optional 3-cycle glitch filters, based on the LPO clock. A detected external pin is required to remain asserted until the enabled glitch filter times out. Additional latency of up to 2 cycles is due to synchronization, which results in a total of up to 5 cycles of delay before the detect circuit alerts the system to the wakeup or reset event when the filter function is enabled. Four wakeup detect filters are available for selected external pins. Glitch filtering is not provided on the internal modules.

For internal module interrupts, the WUMEx bit enables the associated module interrupt as a wakeup source.

For internal module DMA requests, the WUDEx bit enables the associated module DMA request as a temporary wakeup source.

16.4.1 VLLS modes

For any wakeup from VLLS, recovery is always via a reset flow and RCM_SRS[WAKEUP] is set indicating the low-leakage mode was active. State retention data is lost and I/O will be restored after PMC_REGSC[ACKISO] has been written.

A VLLS exit event due to $\overline{\text{RESET}}$ pin assertion causes an exit via a system reset. State retention data is lost and the I/O states immediately return to their reset state. The RCM_SRS[WAKEUP] and RCM_SRS[PIN] bits are set and the system executes a reset flow before CPU operation begins with a reset vector fetch.

16.4.2 Initialization

For an enabled peripheral wakeup input, the peripheral flag must be cleared by software before entering VLLSx mode to avoid an immediate exit from the mode.

Flags associated with external input pins, filtered and unfiltered, must also be cleared by software prior to entry to VLLSx mode.

After enabling an external pin filter or changing the source pin, wait at least five LPO clock cycles before entering VLLSx mode to allow the filter to initialize.

NOTE

After recovering from a VLLS mode, user must restore chip configuration before clearing PMC_REGSC[ACKISO]. In particular, pin configuration for enabled LLWU wake-up pins must be restored to avoid any LLWU flag from being falsely set when PMC_REGSC[ACKISO] is cleared.

The signal selected as a wake-up source pin must be a digital pin, as selected in the pin mux control.

Chapter 17
Reset Control Module (RCM)

17.1 Introduction

Information found here describes the registers of the Reset Control Module (RCM). The RCM implements many of the reset functions for the chip. See the chip's reset chapter for more information.

See AN4503: Power Management for Kinetis MCUs for further details on using the RCM.

17.2 Reset memory map and register descriptions

The RCM Memory Map/Register Definition can be found here.

The Reset Control Module (RCM) registers provide reset status information and reset filter control.

NOTE

The RCM registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

RCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_F000	System Reset Status Register 0 (RCM_SRS0)	8	R	82h	17.2.1/344
4007_F001	System Reset Status Register 1 (RCM_SRS1)	8	R	00h	17.2.2/345
4007_F004	Reset Pin Filter Control register (RCM_RPFC)	8	R/W	00h	17.2.3/346
4007_F005	Reset Pin Filter Width register (RCM_RPFW)	8	R/W	00h	17.2.4/347

17.2.1 System Reset Status Register 0 (RCM_SRS0)

This register includes read-only status flags to indicate the source of the most recent reset. The reset state of these bits depends on what caused the MCU to reset.

NOTE

The reset value of this register depends on the reset source:

- POR (including LVD) — 0x82
- LVD (without POR) — 0x02
- VLLS mode wakeup due to $\overline{\text{RESET}}$ pin assertion — 0x41
- VLLS mode wakeup due to other wakeup sources — 0x01
- Other reset — a bit is set if its corresponding reset source caused the reset

Address: 4007_F000h base + 0h offset = 4007_F000h

Bit	7	6	5	4	3	2	1	0
Read	POR	PIN	WDOG	0		LOC	LVD	WAKEUP
Write								
Reset	1	0	0	0	0	0	1	0

RCM_SRS0 field descriptions

Field	Description
7 POR	Power-On Reset Indicates a reset has been caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0 Reset not caused by POR 1 Reset caused by POR
6 PIN	External Reset Pin Indicates a reset has been caused by an active-low level on the external $\overline{\text{RESET}}$ pin. 0 Reset not caused by external reset pin 1 Reset caused by external reset pin
5 WDOG	Watchdog Indicates a reset has been caused by the watchdog timer timing out. This reset source can be blocked by disabling the watchdog. 0 Reset not caused by watchdog timeout 1 Reset caused by watchdog timeout
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

RCM_SRS0 field descriptions (continued)

Field	Description
2 LOC	Loss-of-Clock Reset Indicates a reset has been caused by a loss of external clock. The MCG clock monitor must be enabled for a loss of clock to be detected. Refer to the detailed MCG description for information on enabling the clock monitor. 0 Reset not caused by a loss of external clock. 1 Reset caused by a loss of external clock.
1 LVD	Low-Voltage Detect Reset If PMC_LVDSC1[LVDRE] is set and the supply drops below the LVD trip voltage, an LVD reset occurs. This field is also set by POR. 0 Reset not caused by LVD trip or POR 1 Reset caused by LVD trip or POR
0 WAKEUP	Low Leakage Wakeup Reset Indicates a reset has been caused by an enabled LLWU module wakeup source while the chip was in a low leakage mode. Any enabled wakeup source in a VLLSx mode causes a reset. This bit is cleared by any reset except WAKEUP. 0 Reset not caused by LLWU module wakeup source 1 Reset caused by LLWU module wakeup source

17.2.2 System Reset Status Register 1 (RCM_SRS1)

This register includes read-only status flags to indicate the source of the most recent reset. The reset state of these bits depends on what caused the MCU to reset.

NOTE

The reset value of this register depends on the reset source:

- POR (including LVD) — 0x00
- LVD (without POR) — 0x00
- VLLS mode wakeup — 0x00
- Other reset — a bit is set if its corresponding reset source caused the reset

Address: 4007_F000h base + 1h offset = 4007_F001h

Bit	7	6	5	4	3	2	1	0
Read	0	0	SACKERR	0	MDM_AP	SW	LOCKUP	0
Write								
Reset	0	0	0	0	0	0	0	0

RCM_SRS1 field descriptions

Field	Description
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 read-only field is reserved and always has the value 0.
5 SACKERR	Stop Mode Acknowledge Error Reset Indicates that after an attempt to enter Stop mode, a reset has been caused by a failure of one or more peripherals to acknowledge within approximately one second to enter stop mode. 0 Reset not caused by peripheral failure to acknowledge attempt to enter stop mode 1 Reset caused by peripheral failure to acknowledge attempt to enter stop mode
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 MDM_AP	MDM-AP System Reset Request Indicates a reset has been caused by the host debugger system setting of the System Reset Request bit in the MDM-AP Control Register. 0 Reset not caused by host debugger system setting of the System Reset Request bit 1 Reset caused by host debugger system setting of the System Reset Request bit
2 SW	Software Indicates a reset has been caused by software setting of SYSRESETREQ bit in Application Interrupt and Reset Control Register in the ARM core. 0 Reset not caused by software setting of SYSRESETREQ bit 1 Reset caused by software setting of SYSRESETREQ bit
1 LOCKUP	Core Lockup Indicates a reset has been caused by the ARM core indication of a LOCKUP event. 0 Reset not caused by core LOCKUP event 1 Reset caused by core LOCKUP event
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

17.2.3 Reset Pin Filter Control register (RCM_RPFC)

NOTE

The reset values of bits 2-0 are for Chip POR only. They are unaffected by other reset types.

NOTE

The bus clock filter is reset when disabled or when entering stop mode. The LPO filter is reset when disabled .

Address: 4007_F000h base + 4h offset = 4007_F004h

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

RCM_RPFC field descriptions

Field	Description
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 RSTFLTSS	Reset Pin Filter Select in Stop Mode Selects how the reset pin filter is enabled in Stop and VLPS modes , and also during VLLS mode. On exit from VLLS mode, this bit should be reconfigured before clearing PMC_REGSC[ACKISO]. 0 All filtering disabled 1 LPO clock filter enabled
RSTFLTSRW	Reset Pin Filter Select in Run and Wait Modes Selects how the reset pin filter is enabled in run and wait modes. 00 All filtering disabled 01 Bus clock filter enabled for normal operation 10 LPO clock filter enabled for normal operation 11 Reserved

17.2.4 Reset Pin Filter Width register (RCM_RPFW)

NOTE

The reset values of the bits in the RSTFLTSEL field are for Chip POR only. They are unaffected by other reset types.

Address: 4007_F000h base + 5h offset = 4007_F005h

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

RCM_RPFW field descriptions

Field	Description
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RSTFLTSEL	Reset Pin Filter Bus Clock Select Selects the reset pin bus clock filter width.

Table continues on the next page...

RCM_RPFW field descriptions (continued)

Field	Description
	00000 Bus clock filter count is 1 00001 Bus clock filter count is 2 00010 Bus clock filter count is 3 00011 Bus clock filter count is 4 00100 Bus clock filter count is 5 00101 Bus clock filter count is 6 00110 Bus clock filter count is 7 00111 Bus clock filter count is 8 01000 Bus clock filter count is 9 01001 Bus clock filter count is 10 01010 Bus clock filter count is 11 01011 Bus clock filter count is 12 01100 Bus clock filter count is 13 01101 Bus clock filter count is 14 01110 Bus clock filter count is 15 01111 Bus clock filter count is 16 10000 Bus clock filter count is 17 10001 Bus clock filter count is 18 10010 Bus clock filter count is 19 10011 Bus clock filter count is 20 10100 Bus clock filter count is 21 10101 Bus clock filter count is 22 10110 Bus clock filter count is 23 10111 Bus clock filter count is 24 11000 Bus clock filter count is 25 11001 Bus clock filter count is 26 11010 Bus clock filter count is 27 11011 Bus clock filter count is 28 11100 Bus clock filter count is 29 11101 Bus clock filter count is 30 11110 Bus clock filter count is 31 11111 Bus clock filter count is 32

Chapter 18
Bit Manipulation Engine (BME)

18.1 Introduction

The Bit Manipulation Engine (BME) provides hardware support for atomic read-modify-write memory operations to the peripheral address space in Cortex-M0+ based microcontrollers.

This architectural capability is also known as "decorated storage" as it defines a mechanism for providing additional semantics for load and store operations to memory-mapped peripherals beyond just the reading and writing of data values to the addressed memory locations. In the BME definition, the "decoration", that is, the additional semantic information, is encoded into the peripheral address used to reference the memory.

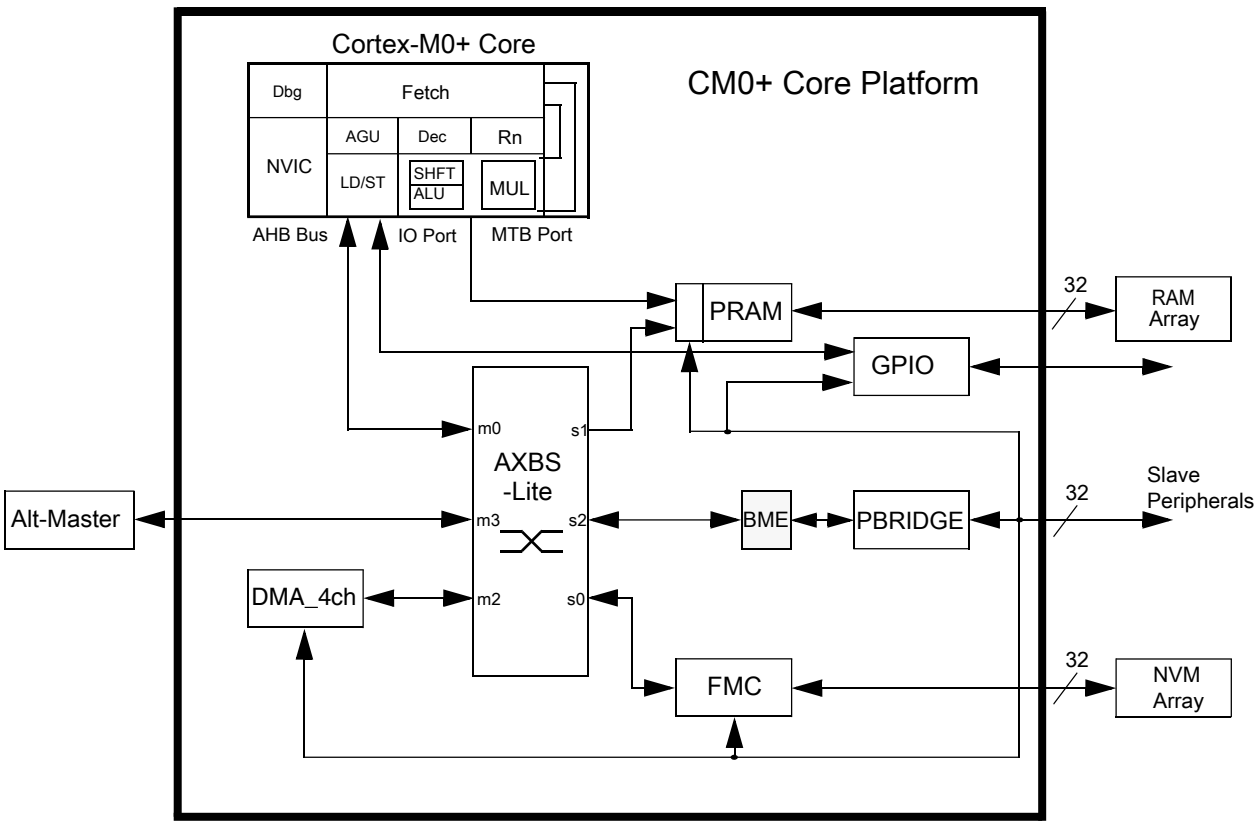
By combining the basic load and store instructions of the ARM Cortex-M instruction set architecture (v6M, v7M) with the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. The resulting architectural capability defined by this core platform function is targeted at the manipulation of n-bit fields in peripheral registers and is consistent with I/O hardware addressing in the Embedded C standard. For most BME commands, a single core read or write bus cycle is converted into an atomic read-modify-write, that is, an indivisible "read followed by a write" bus sequence.

BME decorated references are only available on system bus transactions generated by the processor core and targeted at the standard 512 KB peripheral address space based at 0x4000_0000[1]. The decoration semantic is embedded into address bits[28:19], creating a 448 MB space at addresses 0x4400_0000–0x5FFF_FFFF for AIPS; these bits are stripped out of the actual address sent to the peripheral bus controller and used by the BME to define and control its operation.

1. To be perfectly accurate, the peripheral address space occupies a 516 KB region: 512 KB based at 0x4000_0000 plus a 4 KB space based at 0x400F_F000 for GPIO accesses. This organization provides compatibility with the Kinetis K Family. Attempted accesses to the memory space located between 0x4008_0000 - 0x400F_EFFF are error terminated due to an illegal address.

18.1.1 Overview

The following figure is a generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers.



Note: BME can be accessed only by the core.

Figure 18-1. Cortex-M0+ core platform block diagram

As shown in the block diagram, the BME module interfaces to a crossbar switch AHB slave port as its primary input and sources an AHB bus output to the Peripheral Bridge (PBRIDGE) controller. The BME hardware microarchitecture is a 2-stage pipeline design matching the protocol of the AMBA-AHB system bus interfaces. The PBRIDGE module converts the AHB system bus protocol into the IPS/APB protocol used by the attached slave peripherals.

18.1.2 Features

The key features of the BME include:

- Lightweight implementation of decorated storage for selected address spaces

- Additional access semantics encoded into the reference address
- Resides between a crossbar switch slave port and a peripheral bridge bus controller
- Two-stage pipeline design matching the AHB system bus protocol
- Combinationally passes non-decorated accesses to peripheral bridge bus controller
- Conversion of decorated loads and stores from processor core into atomic read-modify-writes
- Decorated loads support unsigned bit field extracts, load-and-{set,clear} 1-bit operations
- Decorated stores support bit field inserts, logical AND, OR, and XOR operations
- Support for byte, halfword and word-sized decorated operations
- Supports minimum signal toggling on AHB output bus to reduce power dissipation

18.1.3 Modes of operation

The BME module does not support any special modes of operation. As a memory-mapped device located on a crossbar slave AHB system bus port, BME responds strictly on the basis of memory addresses for accesses to the peripheral bridge bus controller.

All functionality associated with the BME module resides in the core platform's clock domain; this includes its connections with the crossbar slave port and the PBRIDGE bus controller.

18.2 Memory map and register definition

The BME module provides a memory-mapped capability and does not include any programming model registers.

The exact set of functions supported by the BME are detailed in the Functional description.

The peripheral address space occupies a 516 KB region: 512 KB based at 0x4000_0000 plus a 4 KB space based at 0x400F_F000 for GPIO accesses; the decorated address space is mapped to the 448 MB region located at 0x4400_0000–0x5FFF_FFFF.

18.3 Functional description

Information found here details the specific functions supported by the BME.

Recall the combination of the basic load and store instructions of the Cortex-M instruction set architecture (v6M, v7M) plus the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. The resulting architectural capability defined by this core platform function is targeted at the manipulation of n-bit fields in peripheral registers and is consistent with I/O hardware addressing in the Embedded C standard. For most BME commands, a single core read or write bus cycle is converted into an atomic read-modify-write, that is, an indivisible "read followed by a write" bus sequence.

Consider decorated store operations first, then decorated loads.

18.3.1 BME decorated stores

The functions supported by the BME's decorated stores include three logical operators (AND, OR, XOR) plus a bit field insert.

For all these operations, BME converts a single decorated AHB store transaction into a 2-cycle atomic read-modify-write sequence, where the combined read-modify operation is performed in the first AHB data phase, and then the write is performed in the second AHB data phase.

A generic timing diagram of a decorated store showing a peripheral bit field insert operation is shown as follows:

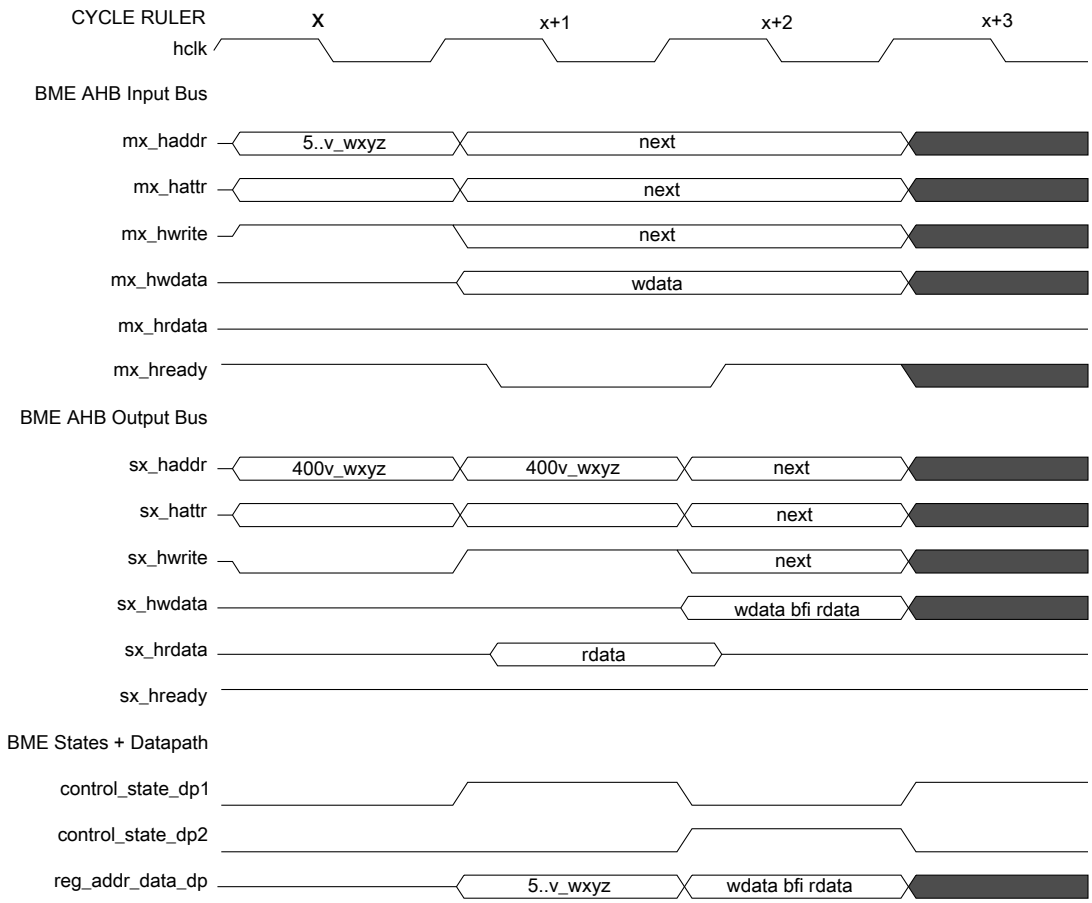


Figure 18-2. Decorated store: bit field insert timing diagram

All the decorated store operations follow the same execution template shown in Figure 18-2, a two-cycle read-modify-write operation:

1. Cycle x, 1st AHB address phase: Write from input bus is translated into a read operation on the output bus using the actual memory address (with the decoration removed) and then captured in a register.
2. Cycle x+1, 2nd AHB address phase: Write access with the registered (but actual) memory address is output
3. Cycle x+1, 1st AHB data phase: Memory read data is modified using the input bus write data and the function defined by the decoration and captured in a data register; the input bus cycle is stalled.
4. Cycle x+2, 2nd AHB data phase: Registered write data is sourced onto the output write data bus.

NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

18.3.1.1 Decorated store logical AND (AND)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read;
2. It is then modified by performing a logical AND operation using the write data operand sourced for the system bus cycle
3. Finally, the result of the AND operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

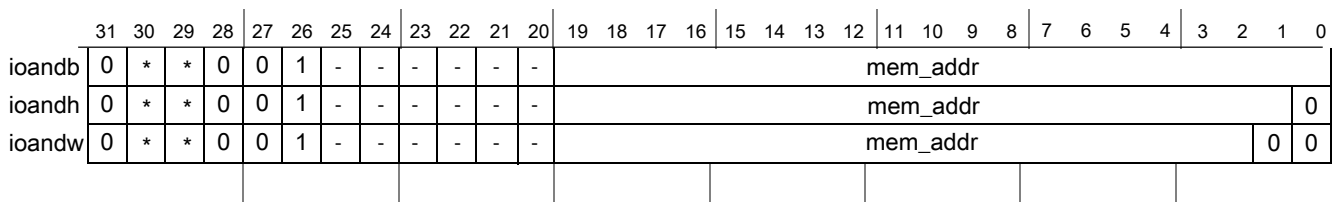


Figure 18-3. Decorated store address: logical AND

See Figure 18-3, where addr[30:29] = 10 for peripheral, addr[28:26] = 001 specifies the AND operation, and mem_addr[19:0] specifies the address offset into the space based at 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care".

The decorated AND write operation is defined in the following pseudo-code as:

```
ioand<sz>(accessAddress, wdata)                  // decorated store AND
tmp   =  mem[accessAddress & 0xE00FFFFF, size]   // memory read
tmp   =  tmp & wdata                             // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp      // memory write
```

where the operand size <sz> is defined as b(yte, 8-bit), h(alfword, 16-bit) and w(ord, 32-bit). This notation is used throughout the document.

In the cycle definition tables, the notations AHB_ap and AHB_dp refer to the address and data phases of the BME AHB transaction. The cycle-by-cycle BME operations are detailed in the following table.

Table 18-1. Cycle definitions of decorated store: logical AND

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert	Recirculate captured addr + attr to memory as slave_wt	<next>

Table continues on the next page...

Table 18-1. Cycle definitions of decorated store: logical AND (continued)

Pipeline stage	Cycle		
	x	x+1	x+2
	master_wt to slave_rd; Capture address, attributes		
BME AHB_dp	<previous>	Perform memory read; Form (rdata & wdata) and capture destination data in register	Perform write sending registered data to memory

18.3.1.2 Decorated store logical OR (OR)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read.
2. It is then modified by performing a logical OR operation using the write data operand sourced for the system bus cycle.
3. Finally, the result of the OR operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

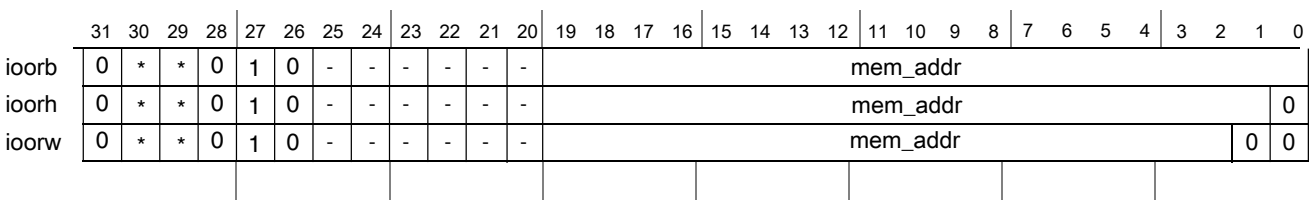


Figure 18-4. Decorated address store: logical OR

See Figure 18-4,where addr[30:29] =10 for peripheral, addr[28:26] = 010 specifies the OR operation, and mem_addr[19:0] specifies the address offset into the space based at 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care".

The decorated OR write operation is defined in the following pseudo-code as:

```
ioor<sz>(accessAddress, wdata)                  // decorated store OR

tmp   =  mem[accessAddress & 0xE00FFFFF, size]  // memory read
tmp   =  tmp | wdata                            // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp     // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 18-2. Cycle definitions of decorated store: logical OR

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form (rdata I wdata) and capture destination data in register	Perform write sending registered data to memory

18.3.1.3 Decorated store logical XOR (XOR)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read.
2. It is then modified by performing a logical XOR (exclusive-OR) operation using the write data operand sourced for the system bus cycle.
3. Finally, the result of the XOR operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

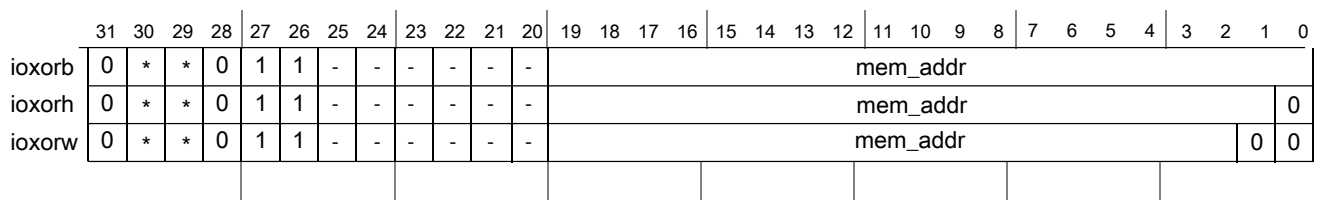


Figure 18-5. Decorated address store: logical XOR

See Figure 18-5, where addr[30:29] =10 for peripheral, addr[28:26] = 011 specifies the XOR operation, and mem_addr[19:0] specifies the address offset into the peripheral space based at 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care".

The decorated XOR write operation is defined in the following pseudo-code as:

```
ioxor<sz>(accessAddress, wdata)                 // decorated store XOR

tmp   =  mem[accessAddress & 0xE00FFFFF, size]  // memory read
tmp   =  tmp ^ wdata                            // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp     // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 18-3. Cycle definitions of decorated store: logical XOR

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form (rdata ^ wdata) and capture destination data in register	Perform write sending registered data to memory

18.3.1.4 Decorated store bit field insert (BFI)

This command inserts a bit field contained in the write data operand, defined by LSB position (b) and the bit field width (w+1), into the memory "container" defined by the access size associated with the store instruction using an atomic read-modify-write sequence.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

NOTE

For the word sized operation, the maximum bit field width is 16 bits. The core performs the required write data lane replication on byte and halfword transfers.

The BFI operation can be used to insert a single bit into a peripheral. For this case, the w field is simply set to 0, indicating a bit field width of 1.

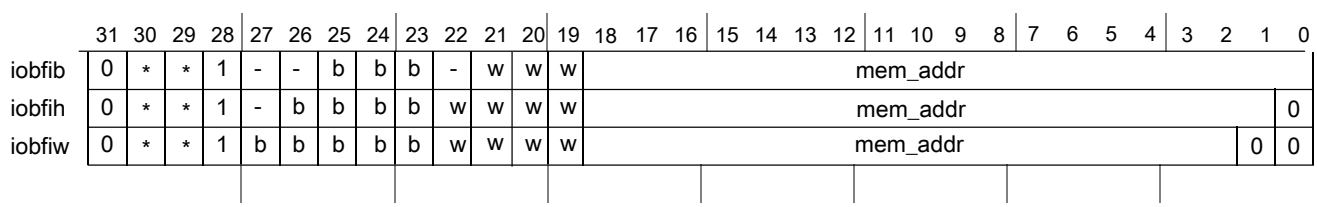


Figure 18-6. Decorated address store: bit field insert

where addr[30:29] =10 for peripheral,addr[28] = 1 signals a BFI operation, addr[27:23] is "b", the LSB identifier, addr[22:19] is "w", the bit field width minus 1 identifier, and addr[18:0] specifies the address offset into the peripheral space based at 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care". Note, unlike the other decorated store operations, BFI uses addr[19] as the least significant bit in the "w" specifier and not as an address bit.

The decorated BFI write operation is defined in the following pseudo-code as:

```
iobfi<sz>(accessAddress, wdata)                  // decorated bit field insert

tmp   =  mem[accessAddress & 0xE007FFFF, size]  // memory read
mask  = ((1 << (w+1)) - 1) << b                 // generate bit mask
tmp   =  tmp   & ~mask                          // modify
      |  wdata &  mask
mem[accessAddress & 0xE007FFFF, size] = tmp     // memory write
```

The write data operand (wdata) associated with the store instruction contains the bit field to be inserted. It must be properly aligned within a right-aligned container, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword, or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field "xyz" into an 8-bit memory container, initially set to "abcd_efgh". For all cases, w is 2, signaling a bit field width of 3.

```
if b = 0 and the decorated store (strb) Rt register[7:0] = ----_-xyz,
   then destination is "abcd_exyz"
if b = 1 and the decorated store (strb) Rt register[7:0] = ----_xyz-,
   then destination is "abcd_xyzh"
if b = 2 and the decorated store (strb) Rt register[7:0] = ---x_yz--,
   then destination is "abcx_yzgh"
if b = 3 and the decorated store (strb) Rt register[7:0] = --xy_z---,
   then destination is "abxy_zfgh"
if b = 4 and the decorated store (strb) Rt register[7:0] = -xyz_----,
   then destination is "axyz_efgh"
if b = 5 and the decorated store (strb) Rt register[7:0] = xyz-_----,
   then destination is "xyzd_efgh"
if b = 6 and the decorated store (strb) Rt register[7:0] = yz--_----,
   then destination is "yzcd_efgh"
if b = 7 and the decorated store (strb) Rt register[7:0] = z---_----,
   then destination is "zbcd_efgh"
```

Note from the example, when the starting bit position plus the field width exceeds the container size, only part of the source bit field is inserted into the destination memory location. Stated differently, if (b + w+1) > container_width, only the low-order "container_width - b" bits are actually inserted.

The cycle-by-cycle BME operations are detailed in the following table.

Table 18-4. Cycle definitions of decorated store: bit field insert

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Form bitwise ((mask) ? wdata : rdata)) and capture destination data in register	Perform write sending registered data to memory

18.3.2 BME decorated loads

The functions supported by the BME's decorated loads include two single-bit load-and-{set, clear} operators plus unsigned bit field extracts.

For the two load-and-{set, clear} operations, BME converts a single decorated AHB load transaction into a two-cycle atomic read-modify-write sequence, where the combined read-modify operations are performed in the first AHB data phase, and then the write is performed in the second AHB data phase as the original read data is returned to the processor core. For an unsigned bit field extract, the decorated load transaction is stalled for one cycle in the BME as the data field is extracted, then aligned and returned to the processor in the second AHB data phase. This is the only decorated transaction that is not an atomic read-modify-write, as it is a simple data read.

A generic timing diagram of a decorated load showing a peripheral load-and-set 1-bit operation is shown as follows.

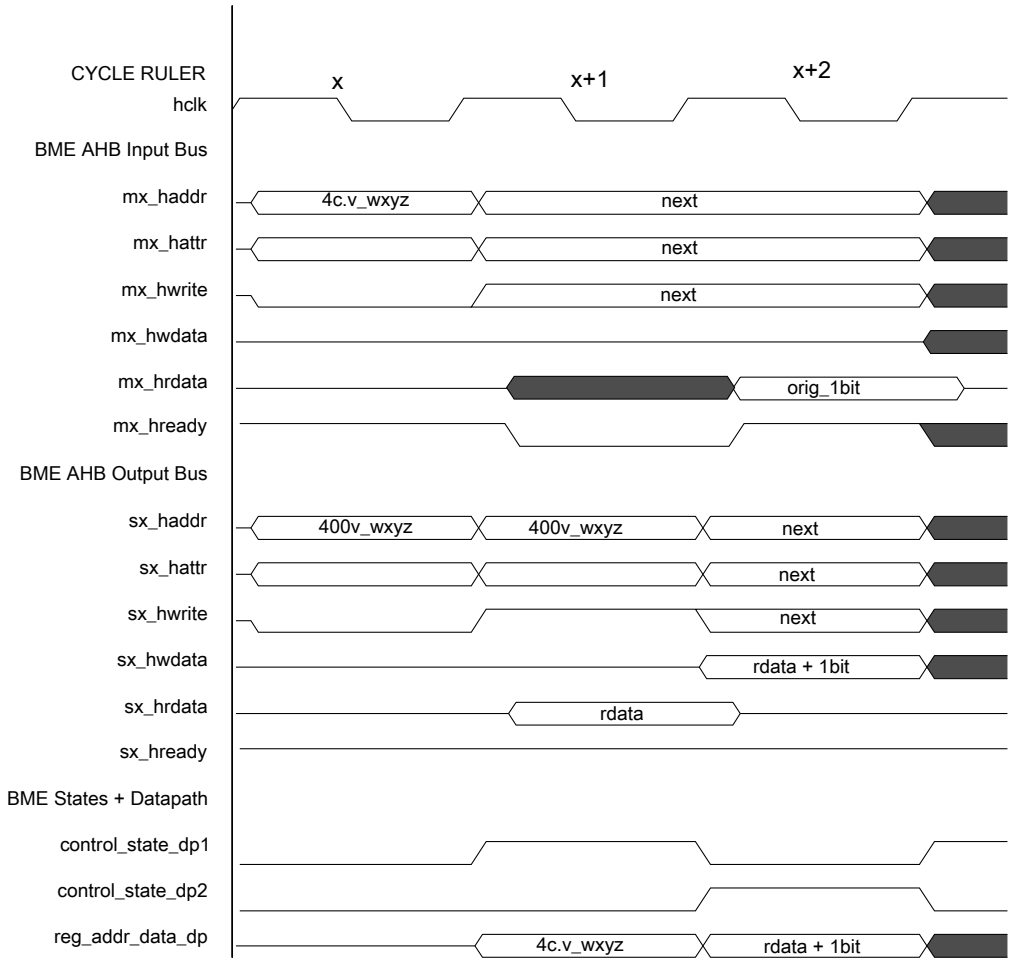


Figure 18-7. Decorated load: load-and-set 1-bit field insert timing diagram

Decorated load-and-{set, clear} 1-bit operations follow the execution template shown in the above figure: a 2-cycle read-modify-write operation:

1. Cycle x, first AHB address phase: Read from input bus is translated into a read operation on the output bus with the actual memory address (with the decoration removed) and then captured in a register
2. Cycle x+1, second AHB address phase: Write access with the registered (but actual) memory address is output
3. Cycle x+1, first AHB data phase: The "original" 1-bit memory read data is captured in a register, while the 1-bit field is set or clear based on the function defined by the decoration with the modified data captured in a register; the input bus cycle is stalled
4. Cycle x+2, second AHB data phase: The selected original 1-bit is right-justified, zero-filled and then driven onto the input read data bus, while the registered write data is sourced onto the output write data bus

NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

A generic timing diagram of a decorated load showing an unsigned peripheral bit field operation is shown in the following figure.

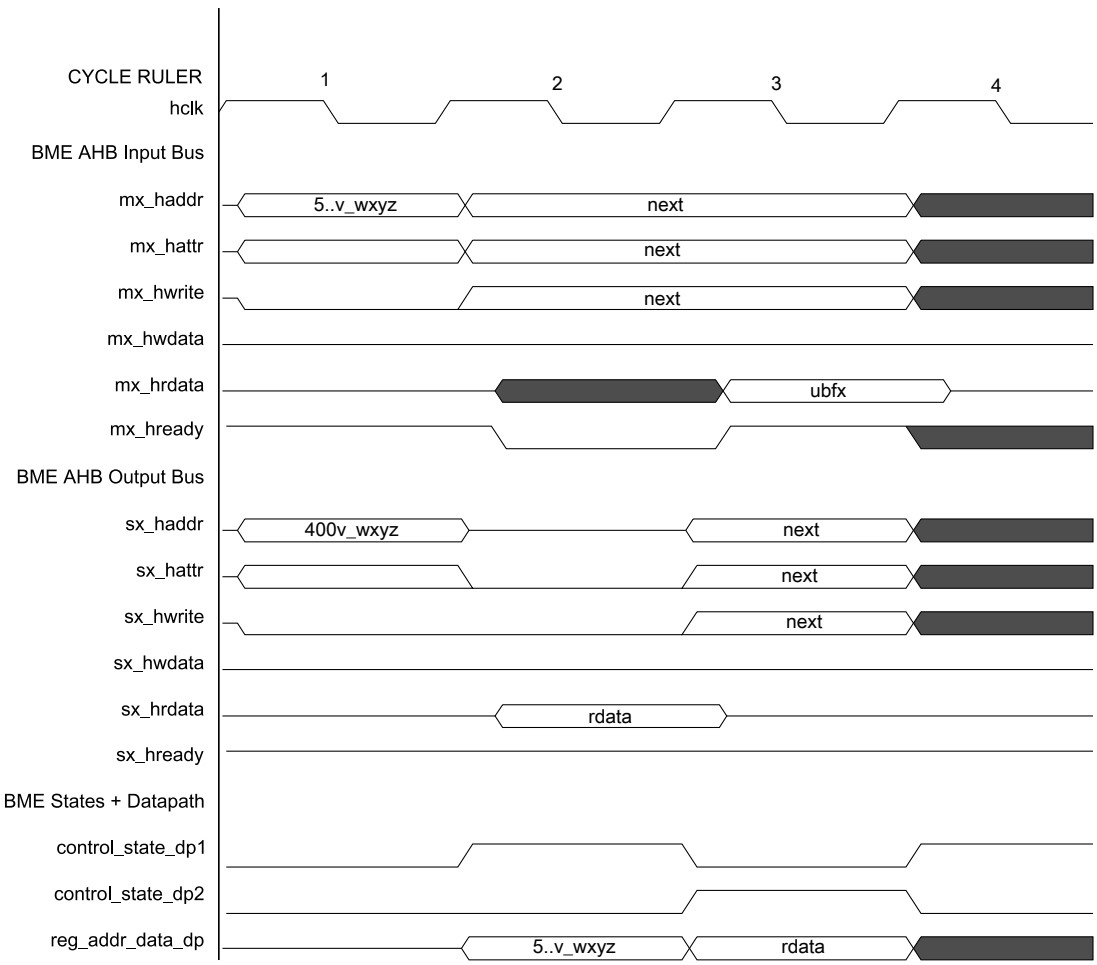


Figure 18-8. Decorated load: unsigned bit field insert timing diagram

The decorated unsigned bit field extract follows the same execution template shown in the above figure, a 2-cycle read operation:

- Cycle x, 1st AHB address phase: Read from input bus is translated into a read operation on the output bus with the actual memory address (with the decoration removed) and then captured in a register
- Cycle x+1, 2nd AHB address phase: Idle cycle

- Cycle x+1, 1st AHB data phase: A bit mask is generated based on the starting bit position and the field width; the mask is AND'ed with the memory read data to isolate the bit field; the resulting data is captured in a data register; the input bus cycle is stalled
- Cycle x+2, 2nd AHB data phase: Registered data is logically right-aligned for proper alignment and driven onto the input read data bus

NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

18.3.2.1 Decorated load: load-and-clear 1 bit (LAC1)

This command loads a 1-bit field defined by the LSB position (b) into the core's general purpose destination register (Rt) and zeroes the bit in the memory space after performing an atomic read-modify-write sequence.

The extracted 1-bit data field from the memory address is right-justified and zero-filled in the operand returned to the core.

The data size is specified by the read operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

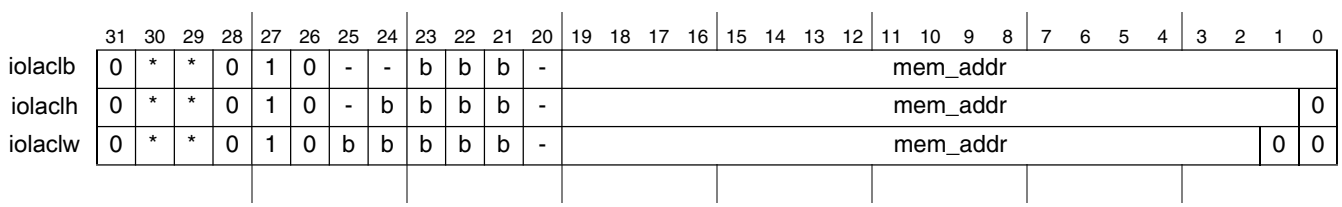


Figure 18-9. Decorated load address: load-and-clear 1 bit

See Figure 18-9, where addr[30:29] = 10 for peripheral, addr[28:26] = 010 specifies the load-and-clear 1 bit operation, addr[25:21] is "b", the bit identifier, and mem_addr[19:0] specifies the address offset into the space based at 0x4000_0000 for peripheral. The "-" indicates an address bit "don't care".

The decorated load-and-clear 1-bit read operation is defined in the following pseudo-code as:

```
rdata =  iolac1<sz>(accessAddress)              // decorated load-and-clear 1

tmp   =  mem[accessAddress & 0xE00FFFFF, size]  // memory read
mask  =  1 << b                                 // generate bit mask
rdata = (tmp &  mask) >> b                      // read data returned to core
tmp   =  tmp & ~mask                            // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp     // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 18-5. Cycle definitions of decorated load: load-and-clear 1 bit

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Extract bit from rdata; Form (rdata & ~mask) and capture destination data in register	Return extracted bit to master; Perform write sending registered data to memory

18.3.2.2 Decorated Load: Load-and-Set 1 Bit (LAS1)

This command loads a 1-bit field defined by the LSB position (b) into the core's general purpose destination register (Rt) and sets the bit in the memory space after performing an atomic read-modify-write sequence.

The extracted one bit data field from the memory address is right justified and zero filled in the operand returned to the core.

The data size is specified by the read operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

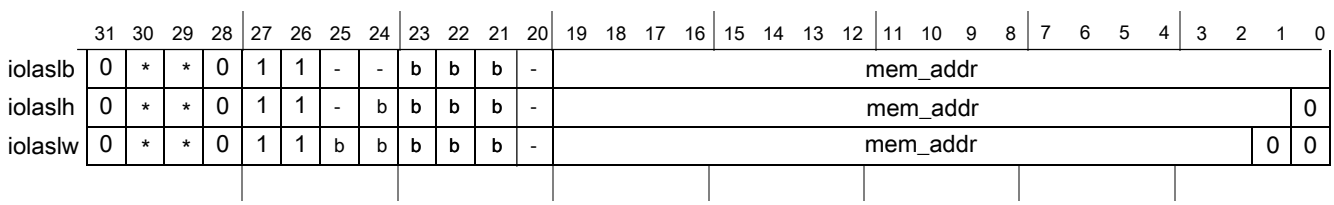


Figure 18-10. Decorated load address: load-and-set 1 bit

where addr[30:29] = 10 for peripheral, addr[28:26] = 011 specifies the load-and-set 1 bit operation, addr[25:21] is "b", the bit identifier, and mem_addr[19:0] specifies the address offset into the space based at 0x4000_0000 for peripheral. The "-" indicates an address bit "don't care".

The decorated Load-and-Set 1 Bit read operation is defined in the following pseudo-code as:

```
rdata =  iolas1<sz>(accessAddress)              // decorated load-and-set 1

tmp   =  mem[accessAddress & 0xE00FFFFF, size]  // memory read
mask  =  1 << b                                 // generate bit mask
rdata = (tmp &  mask) >> b                      // read data returned to core
```

```
tmp    =  tmp |  mask                              // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp       // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 18-6. Cycle definitions of decorated load: load-and-set 1-bit

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Extract bit from rdata; Form (rdata | mask) and capture destination data in register	Return extracted bit to master; Perform write sending registered data to memory

18.3.2.3 Decorated load unsigned bit field extract (UBFX)

This command extracts a bit field defined by LSB position (b) and the bit field width (w +1) from the memory "container" defined by the access size associated with the load instruction using a two-cycle read sequence.

The extracted bit field from the memory address is right-justified and zero-filled in the operand returned to the core. Recall this is the only decorated operation that does not perform a memory write, that is, UBFX only performs a read.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). Note for the word sized operation, the maximum bit field width is 16 bits.

The use of a UBFX operation is recommended to extract a single bit. For this case, the w field is simply set to 0, indicating a bit field width of 1.

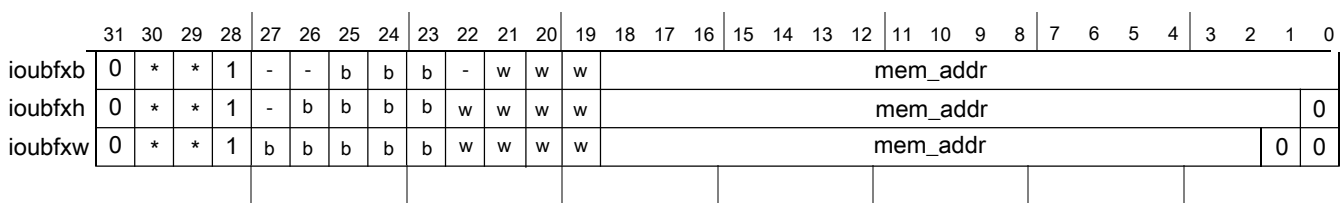


Figure 18-11. Decorated load address: unsigned bit field extract

See Figure 18-11, where addr[30:29] = 10 for peripheral, addr[28] = 1 specifies the unsigned bit field extract operation, addr[27:23] is "b", the LSB identifier, addr[22:19] is "w", the bit field width minus 1 identifier, and mem_addr[18:0] specifies the address

offset into the space based at 0x4000_0000 for peripheral. The "-" indicates an address bit "don't care". Note, unlike the other decorated load operations, UBFX uses addr[19] as the least significant bit in the "w" specifier and not as an address bit.

The decorated unsigned bit field extract read operation is defined in the following pseudo-code as:

```
rdata =  ioubfx<sz>(accessAddress)              // unsigned bit field extract

tmp   =  mem[accessAddress & 0xE007FFFF, size]  // memory read
mask  = ((1 << (w+1)) - 1) << b                 // generate bit mask
rdata = (tmp &  mask) >> b                      // read data returned to core
```

Like the BFI operation, when the starting bit position plus the field width exceeds the container size, only part of the source bit field is extracted from the destination memory location. Stated differently, if (b + w+1) > container_width, only the low-order "container_width - b" bits are actually extracted. The cycle-by-cycle BME operations are detailed in the following table.

Table 18-7. Cycle definitions of decorated load: unsigned bit field extract

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Idle AHB address phase	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Form (rdata & mask) and capture destination data in register	Logically right shift registered data; Return justified rdata to master

18.3.3 Additional details on decorated addresses and GPIO accesses

As previously noted, the peripheral address space occupies a 516 KB region: 512 KB based at 0x4000_0000 plus a 4 KB space based at 0x400F_F000 for GPIO accesses. This memory layout provides compatibility with the Kinetis K Family and provides 129 address "slots", each 4 KB in size.

The GPIO address space is multiply-mapped by the hardware: it appears at the "standard" system address 0x400F_F000 and is physically located in the address slot corresponding to address 0x4000_F000. Decorated loads and stores create a slight complication involving accesses to the GPIO. Recall the use of address[19] varies by decorated operation; for AND, OR, XOR, LAC1 and LAS1, this bit functions as a true address bit, while for BFI and UBFX, this bit defines the least significant bit of the "w" bit field specifier.

As a result, undecorated GPIO references and decorated AND, OR, XOR, LAC1 and LAS1 operations can use the standard 0x400F_F000 base address, while decorated BFI and UBFX operations must use the alternate 0x4000_F000 base address. Another implementation can simply use 0x400F_F000 as the base address for all undecorated GPIO accesses and 0x4000_F000 as the base address for all decorated accesses. Both implementations are supported by the hardware.

Table 18-8. Decorated peripheral and GPIO address details

Peripheral address space	Description
0x4000_0000–0x4007_FFFF	Undecorated (normal) peripheral accesses
0x4008_0000–0x400F_EFFF	Illegal addresses; attempted references are aborted and error terminated
0x400F_F000–0x400F_FFFF	Undecorated (normal) GPIO accesses using standard address
0x4010_0000–0x43FF_FFFF	Illegal addresses; attempted references are aborted and error terminated
0x4400_0000–0x4FFF_FFFF	Decorated AND, OR, XOR, LAC1, LAS1 references to peripherals and GPIO based at either 0x4000_F000 or 0x400F_F000
0x5000_0000–0x5FFF_FFFF	Decorated BFI, UBFX references to peripherals and GPIO only based at 0x4000_F000

18.4 Application information

In this section, GNU assembler macros with C expression operands are presented as examples of the required instructions to perform decorated operations.

This section specifically presents a partial bme.h file defining the assembly language expressions for decorated logical stores: AND, OR, and XOR. Comparable functions for BFI and the decorated loads are more complex and available in the complete BME header file.

These macros use the same function names presented in Functional description.

```
#define IOANDW(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "str    r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOANDH(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "strh   r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOANDB(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "strb   r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
```

```
#define IOORW(ADDR,WDATA)            \
    __asm("ldr    r3, =(1<<27);"     \
          "orr    r3, %[addr];"      \
          "mov    r2, %[wdata];"     \
          "str    r2, [r3];"         \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOORH(ADDR,WDATA)            \
    __asm("ldr    r3, =(1<<27);"     \
          "orr    r3, %[addr];"      \
          "mov    r2, %[wdata];"     \
          "strh   r2, [r3];"         \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOORB(ADDR,WDATA)            \
    __asm("ldr    r3, =(1<<27);"     \
          "orr    r3, %[addr];"      \
          "mov    r2, %[wdata];"     \
          "strb   r2, [r3];"         \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORW(ADDR,WDATA)           \
    __asm("ldr    r3, =(3<<26);"     \
          "orr    r3, %[addr];"      \
          "mov    r2, %[wdata];"     \
          "str    r2, [r3];"         \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORH(ADDR,WDATA)           \
    __asm("ldr    r3, =(3<<26);"     \
          "orr    r3, %[addr];"      \
          "mov    r2, %[wdata];"     \
          "strh   r2, [r3];"         \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORB(ADDR,WDATA)           \
    __asm("ldr    r3, =(3<<26);"     \
          "orr    r3, %[addr];"      \
          "mov    r2, %[wdata];"     \
          "strb   r2, [r3];"         \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
```

Chapter 19
Memory-Mapped Divide and Square Root (MMDVSQ)

19.1 Introduction

ARM processor cores in the Cortex-M family implementing the ARMv6-M instruction set architecture do not include hardware support for integer divide operations. The affected processors include the Cortex-M0+ core. However, in certain deeply embedded application spaces, hardware support for this class of arithmetic operation (along with an unsigned square root function) is important to maximize system performance and minimize device power dissipation. Accordingly, the MMDVSQ module is included in select microcontrollers, to serve as a memory-mapped co-processor located in a special address space (within the system memory map) that is accessible only to the processor core.

The MMDVSQ module supports execution of the integer divide operations defined in the ARMv7-M instruction set architecture, plus an unsigned integer square root operation. The supported integer divide operations include 32/32 signed (SDIV) and unsigned (UDIV) calculations.

19.1.1 Features

The key features of the MMDVSQ include:

- Lightweight implementation of 32-bit integer divide and square root arithmetic operations
  - Supports 32/32 signed and unsigned divide (or remainder) calculations
  - Supports 32-bit unsigned square root calculations
- Simple programming model includes input data and result registers plus a control/status register
- Programming model interface optimized for activation from inline code or software library call

- "Fast Start" configuration minimizes the memory-mapped register write overhead
- Supports two methods to determine when result is valid, including software polling
- Configurable divide-by-zero response

• Pipelined design processes 2 bits per cycle with early termination exit for minimum execution time

19.1.2 Block diagram

A generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers is shown in Figure 19-1. The MMDVSQ module's location as a memory-mapped co-processor is highlighted.

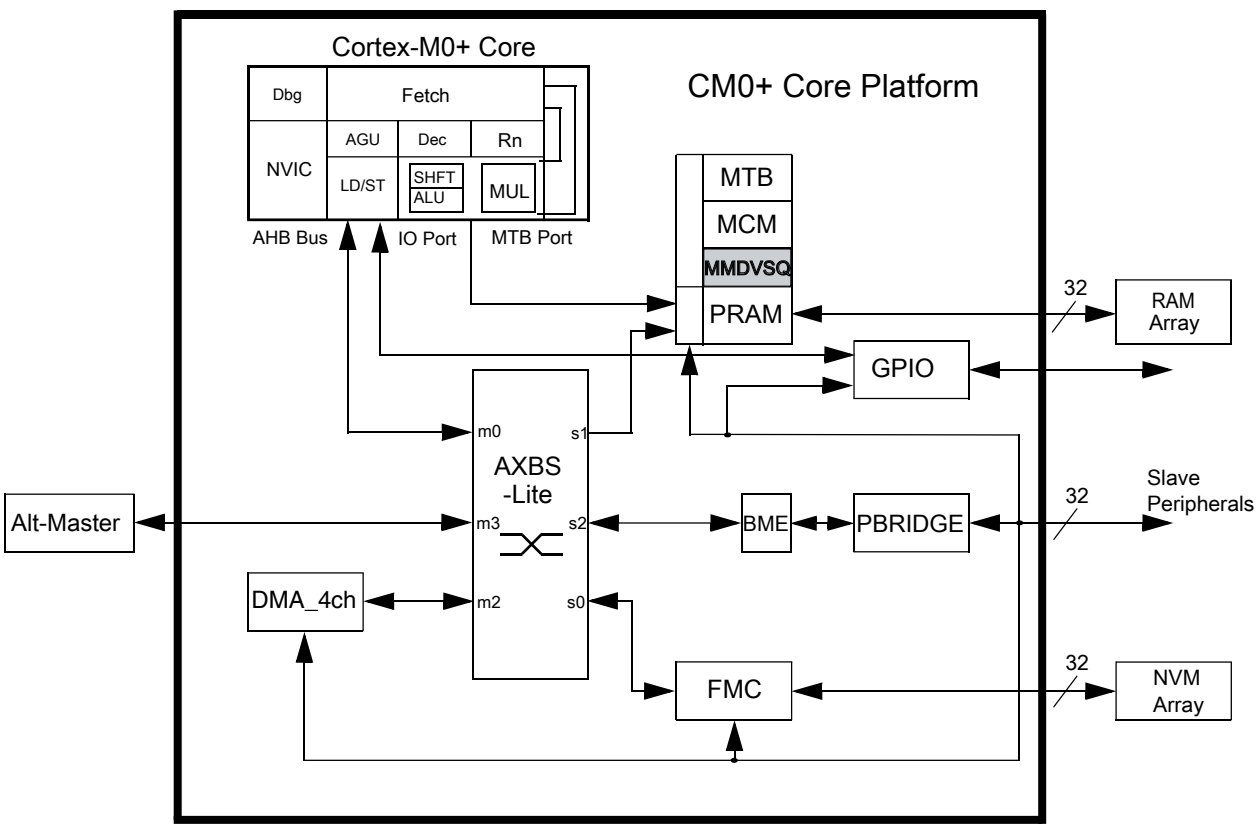


Figure 19-1. Generic Cortex-M0+ Core Platform Block Diagram

Next, a block diagram of the internal structure of the MMDVSQ module is presented. See Figure 19-2.

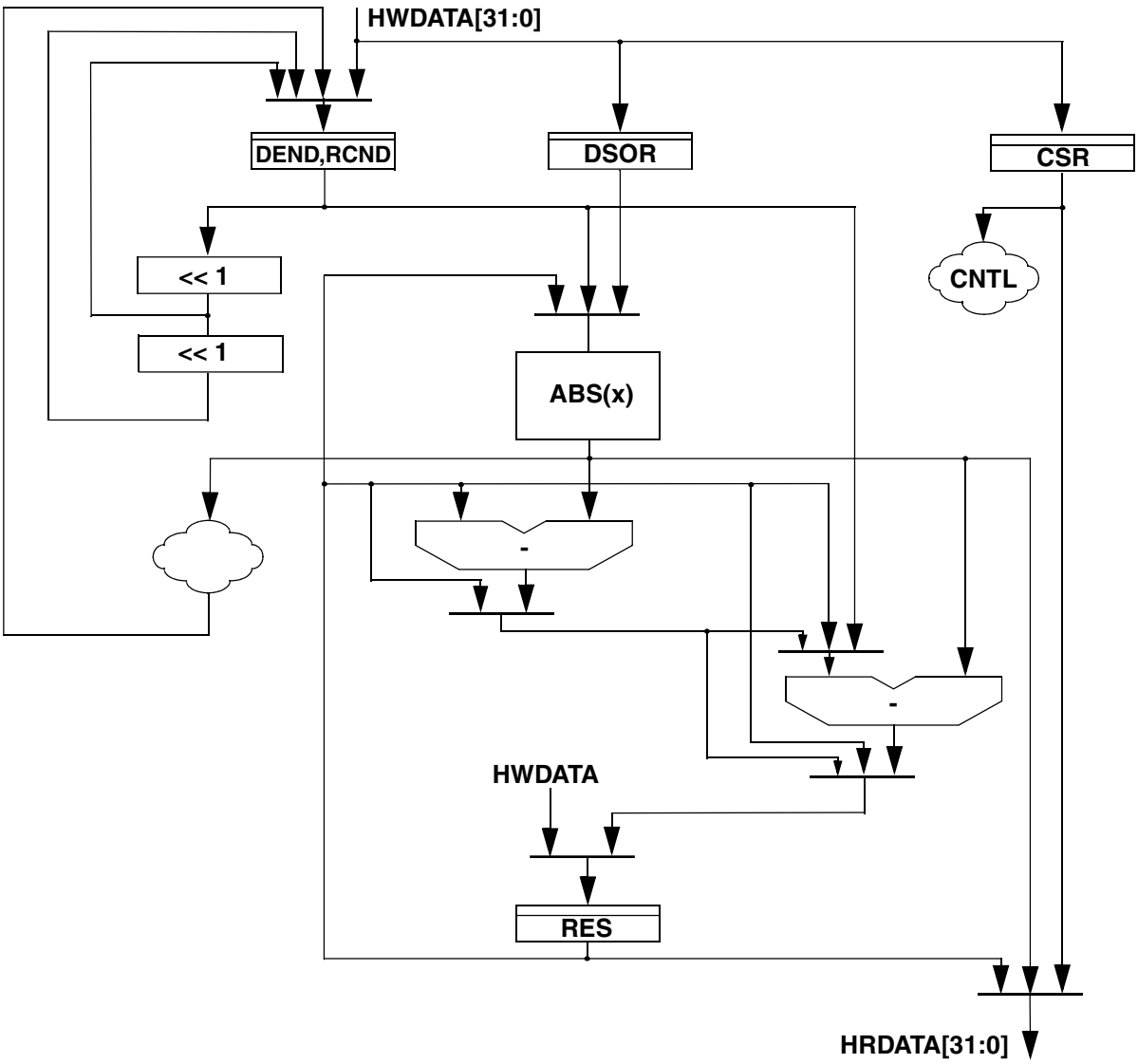


Figure 19-2. MMDVSQ Block Diagram

19.1.3 Modes of operation

The MMDVSQ module does not support any special modes of operation. As a memory-mapped device located on a crossbar slave AHB system bus port, MMDVSQ responds based strictly on memory addresses to its programming model.

All functionality associated with the MMDVSQ module resides in the core platform's clock domain; this includes its connections with the crossbar slave port. To minimize power dissipation, the design supports an architectural clock gate for the entire module, that is, the MMDVSQ is only clocked when responding to bus requests to its programming model or is busy performing a calculation.

19.2 External signal description

The MMDVSQ module does not directly support any external interfaces.

The internal interface includes a standard 32-bit AHB bus as shown in Figure 19-1.

19.3 Memory map and register definition

The MMDVSQ module supports a small number of program-visible registers used for passing input operands and retrieving the output result plus a configuration/status register.

The programming model occupies the first 20 bytes of a standard 4 Kb address slot. It can only be accessed via word-sized (32 bit) accesses. Attempted accesses using smaller data sizes, reading the write-only location or to reserved space are terminated with an error.

At any instant in time, the MMDVSQ can perform either a divide or square root calculation. The basic integer operations supported by the MMDVSQ are:

For divide:

```
MMDVSQ_RES = quotient  (MMDVSQ_DEND / MMDVSQ_DSOR)

MMDVSQ_RES = remainder (MMDVSQ_DEND % MMDVSQ_DSOR)
```

For square root:

```
MMDVSQ_RES = integer   (√MMDVSQ_RCND)
```

The register usage, based on the operation (divide, square root), is detailed in Table 19-1.

Table 19-1. Register Usage = f(Divide, Square Root)

Register	Divide	Square Root	Description
Dividend (MMDVSQ_DEND)	Yes	No	Input dividend (numerator) for the divide
Divisor (MMDVSQ_DSOR)	Yes	No	Input divisor (denominator) for the divide
Control/Status (MMDVSQ_CSR)	Yes	Yes	Control for divide, status for divide and square root
Result (MMDVSQ_RES)	Yes	Yes	Output result
Radicand (MMDVSQ_RCND)	No	Yes	Input "square" data

MMDVSQ memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_4000	Dividend Register (MMDVSQ_DEND)	32	R/W	Undefined	19.3.1/373
F000_4004	Divisor Register (MMDVSQ_DSOR)	32	R/W	Undefined	19.3.2/373
F000_4008	Control/Status Register (MMDVSQ_CSR)	32	R/W	See section	19.3.3/375
F000_400C	Result Register (MMDVSQ_RES)	32	R/W	Undefined	19.3.4/378
F000_4010	Radicand Register (MMDVSQ_RCND)	32	W	Undefined	19.3.5/378

19.3.1 Dividend Register (MMDVSQ_DEND)

This register is loaded with the input dividend operand before a divide operation is initiated. The register is updated by the MMDVSQ hardware during the execution of a divide or square root calculation. Any memory access (read or write) of the DEND register while the module is busy during a calculation causes the access to be stalled (using wait states) until the calculation completes.

Address: F000_4000h base + 0h offset = F000_4000h

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	DIVIDEND																															
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.

MMDVSQ_DEND field descriptions

Field	Description
DIVIDEND	Dividend This is the input dividend operand for divide calculations.

19.3.2 Divisor Register (MMDVSQ_DSOR)

This register is loaded with the input divisor operand before a divide operation is initiated. If CSR[DFS] = 0, a write to this register inititates a divide operation. Any memory access (read or write) of the DSOR register while the module is busy during a calculation causes the access to be stalled (using wait states) until the calculation completes.

If a divide operation is inititated with DSOR = 0, the hardware signals a divide-by-zero condition and sets RES = 0 and CSR[DZ] = 1. If CSR[DZE] = 1, an attempted read of the RES result is error terminated.

Address: F000_4000h base + 4h offset = F000_4004h

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	DIVISOR																															
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.

MMDVSQ_DSOR field descriptions

Field	Description
DIVISOR	Divisor This is the input divisor operand for divide calculations.

19.3.3 Control/Status Register (MMDVSQ_CSR)

This register defines the operating configuration of divide operations and provides status information. The upper 3 bits provide busy status indicators, while the low-order byte defines the configuration for divide operations. The read-only status bits in CSR[31:29] are valid for both divide and square root operations; the configuration and status bit in CSR[5:0] are only valid for divides. A memory write access of the CSR register while the module is busy during a calculation causes the access to be stalled (using wait states) until the calculation completes.

Address: F000_4000h base + 8h offset = F000_4008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUSY	DIV	SQRT	0												
W																
Reset	0	x*	x*	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										DFS	DZ	DZE	REM	USGN	0
W																SRT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- x = Undefined at reset.

MMDVSQ_CSR field descriptions

Field	Description
31 BUSY	BUSY This read-only bit is asserted when the MMDVSQ is performing a divide or square root. When an operation is initiated, the hardware sets this flag. It remains asserted until the operation completes and the

Table continues on the next page...

MMDVSQ_CSR field descriptions (continued)

Field	Description
	hardware automatically clears the indicator. This bit can be used to poll the DVSQ's execution status. The combined CSR[BUSY, DIV, SQRT] indicators provide an encoded module status: • If 0b001, then MMDVSQ is idle and the last calculation was a square root • If 0b010, then MMDVSQ is idle and the last calculation was a divide • If 0b101, then MMDVSQ is busy processing a square root calculation • If 0b110, then MMDVSQ is busy processing a divide calculation The remaining encodings of CSR[BUSY, DIV, SQRT] are reserved. 0 MMDVSQ is idle 1 MMDVSQ is busy performing a divide or square root calculation
30 DIV	DIVIDE Current or last operation was a divide. This read-only indicator bit signals if the current or last operation performed by the MMDVSQ was a divide. 0 Current or last MMDVSQ operation was not a divide 1 Current or last MMDVSQ operation was a divide
29 SQRT	SQUARE ROOT Current or last operation was a square root. This read-only indicator bit signals if the current or last operation performed by the MMDVSQ was a square root. 0 Current or last MMDVSQ operation was not a square root 1 Current or last MMDVSQ operation was a square root
28–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 DFS	Disable Fast Start The MMDVSQ supports 2 mechanisms for initiating a divide operation. The default mechanism is a "fast start" where a write to the DSOR register begins the divide. Alternatively, the start mechanism can begin after a write to the CSR register with CSR[SRT] set. The CSR[DFS] indicator selects the divide start mechanism. 0 A divide operation is initiated by a write to the DSOR register 1 A divide operation is initiated by a write to the CSR register with CSR[SRT] = 1
4 DZ	Divide-by-Zero This read-only status indicator signals the last divide operation had a zero divisor, that is, DSOR = 0x0000_0000. For this case, RES is set to 0x0000_0000 and this indicator bit set. After a divide-by-zero operation, a read of the RES register returns either the zero result, or, if CSR[DZE] = 1, terminates the read with an error. The CSR[DZ] indicator is cleared by the hardware at the beginning of each operation. 0 The last divide operation had a non-zero divisor, that is, DSOR != 0 1 The last divide operation had a zero divisor, that is, DSOR = 0
3 DZE	Divide-by-Zero-Enable This indicator configures the MMDVSQ's response to divide-by-zero calculations. If both CSR[DZ] and CSR[DZE] are set, then a subsequent read of the RES register is error terminated to signal the processor of the attempted divide-by-zero.

Table continues on the next page...

MMDVSQ_CSR field descriptions (continued)

Field	Description
	0 Reads of the RES register return the register contents 1 If CSR[DZ] = 1, an attempted read of RES register is error terminated to signal a divide-by-zero, else the register contents are returned
2 REM	REMainder calculation This indicator selects whether the quotient or the remainder is returned in the RES register. The combined CSR[REM] and CSR[USGN] bits define four possible divide operations: • If CSR[REM, USGN] = 0b00, perform a signed divide, returning the quotient • If CSR[REM, USGN] = 0b01, perform an unsigned divide, returning the quotient • If CSR[REM, USGN] = 0b10, perform a signed divide, returning the remainder • If CSR[REM, USGN] = 0b11, perform an unsigned divide, returning the remainder 0 Return the quotient in the RES for the divide calculation 1 Return the remainder in the RES for the divide calculation
1 USGN	Unsigned calculation This indicator selects whether a signed (default) or unsigned divide is performed. See the CSR[REM] description for the encoding of the four possible divide operations. 0 Perform a signed divide 1 Perform an unsigned divide
0 SRT	Start When written with a logical one and CSR[DFS] = 1, this flag initiates a divide operation. If written as a logical one with CSR[DFS] = 0, it is ignored. This bit always reads as a zero. The state of the register write data defines this bit's function. 0 No operation initiated 1 If CSR[DFS] = 1, then initiate a divide calculation, else ignore

19.3.4 Result Register (MMDVSQ_RES)

This register is loaded with the result of the divide or square root calculation. It is updated by the MMDVSQ hardware at the completion of the calculation. When a square root operation is performed (on an unsigned 32-bit number), the result is limited to a 16-bit value with RES[31:16] = 0x0000. Any memory access (read or write) of the RES register while the module is busy during a calculation causes the access to be stalled (using wait states) until the calculation completes and the new result written into the register.

Address: F000_4000h base + Ch offset = F000_400Ch

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	RESULT																															
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.

MMDVSQ_RES field descriptions

Field	Description
RESULT	Result This is the output result for a divide or square root calculation.

19.3.5 Radicand Register (MMDVSQ_RCND)

The write-only radicand register is loaded with the input “square” number. A memory write to the radicand register initiates a square root calculation. While the MMDVSQ module is busy performing a square root calculation, any memory write access to the RCND register causes the write access to be stalled (using wait states) until the square root calculation finishes. Any attempted read of the radicand register terminates with an error.

Address: F000_4000h base + 10h offset = F000_4010h

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	RADICAND																															
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.

MMDVSQ_RCND field descriptions

Field	Description
RADICAND	Radicand This is the input radicand for a square root calculation, that is, the input "square" number.

19.4 Functional description

This section details the algorithms, execution times of the MMDVSQ, and the software interface to the module.

19.4.1 Algorithms

This section provides more details on the integer divide and square root algorithms.

19.4.1.1 Integer divide including special cases

19.4.1.1.1 Overview

The MMDVSQ module implements a "shift, test, and restore" radix-2 algorithm for unsigned integer divide operations. When performing a signed divide calculation, negative input operands are converted into 2's complement positive numbers first, an unsigned divide performed, and the sign of the results based on the input operand signs, namely:

- The sign of the remainder is the same as the sign of the dividend
- The quotient is negated if the signs of the dividend and divisor are different

The hardware implementation processes two bits per machine cycle and includes "early termination" logic where the execution time is data dependent, based on the magnitude of the positive dividend. See Table 19-4 for more execution time details.

19.4.1.1.2 Special case: Overflow

There is a single "special overflow case" affecting signed integer divides. If the dividend = 0x8000_0000 and the divisor = 0xFFFF_FFFF, the result of this ($-2^{31}/-1$) operation cannot be expressed as a 32-bit 2's complement number. For this case, the MMDVSQ exactly follows the ARM Cortex-Mx definition and returns 0x8000_0000 (the lower 32 bits of the $+2^{31}$ result) as the quotient with no indication of the overflow condition. If the remainder is selected as the output of this calculation, it returns 0x0000_0000.

19.4.1.1.3 Special case: Divide-by-Zero

For both signed and unsigned divides, if the divisor is zero, the MMDVSQ hardware detects this condition and the CSR[DZ] indicator set. The quotient result is forced to 0x0000_0000. If the remainder is selected as the output of this calculation, it also returns 0x0000_0000. Additionally, if CSR[DZE] = 1, then an attempted read of the Result register (RES) is error terminated to provide a simple mechanism to signal software of the divide-by-zero condition.

19.4.1.2 Integer square root

19.4.1.2.1 Overview

The unsigned square root algorithm begins by creating a 32-bit “one-hot” bit vector signaling the highest power of four of the contents of the Radicand register (RCND). It then iterates through an algorithm involving magnitude comparisons of the RCND register versus the working result plus bit vector summation, conditional decrementing of the radicand, a 1-bit right shift of the result, and a 2-bit right shift of the one-hot bit vector.

Processing two bits of the radicand per cycle, the result register finishes with the integer portion of the square root calculation. The module includes early termination logic so that the execution time is data dependent, based on the magnitude of the input radicand. See Table 19-5 for more execution time details. Since both algorithms share common hardware structures, the incremental cost of the square root logic is an extremely small delta to the basic divide hardware.

The square root algorithm was exhaustively compared (that is, all 2^{32} possible input values) against the standard GNU C library implementation, which converts the unsigned integer input into a double-precision floating-point number, calculates the double-precision square root and then converts it back into an unsigned integer. Each input value calculated identical square root results.

19.4.1.2.2 Square root using Q notation

Consider the use of Q notation for square root calculations returning fractional values. The following description is taken from http://en.wikipedia.org/wiki/Q_(number_format).

Q is a fixed point number format where the number of fractional bits (and optionally the number of integer bits) is specified. For example, a Q15 number has 15 fractional bits; a Q1.14 number has 1 integer bit and 14 fractional bits. Q format is often used in hardware that does not have a floating-point unit and in applications that require constant resolution.

Q format numbers are (notionally) fixed point numbers (but not actually a number itself); that is, they are stored and operated upon as regular binary numbers (i.e. signed integers), thus allowing standard integer hardware/ALU to perform rational number calculations. The number of integer bits, fractional bits and the underlying word size are to be chosen by the programmer on an application-specific basis - the programmer's choices of the foregoing will depend on the range and resolution needed for the numbers. The machine itself remains oblivious to the notional fixed point representation being employed - it merely performs integer arithmetic the way it knows how. Ensuring that the computational results are valid in the Q format representation is the responsibility of the programmer.

The Q notation is written as Qm.n, where:

- *Q designates that the number is in the Q format notation - the Texas Instruments representation for signed fixed-point numbers (the "Q" being reminiscent of the standard symbol for the set of rational numbers).*
- *m is the number of bits set aside to designate the two's complement integer portion of the number, exclusive of the sign bit (therefore if m is not specified it is taken as zero).*
- *n is the number of bits used to designate the fractional portion of the number, i.e. the number of bits to the right of the binary point. (If n = 0, the Q numbers are integers - the degenerate case).*

Note that the most significant bit is always designated as the sign bit (the number is stored as a two's complement number) in order to allow standard arithmetic-logic hardware to manipulate Q numbers. Representing a signed fixed-point data type in Q format therefore always requires m+n+1 bits to account for the sign bit. Hence the smallest machine word size required to accommodate a Qm.n number is m+n+1, with the Q number left justified in the machine word.

For a given Qm.n format, using an m+n+1 bit signed integer container with n fractional bits:

- *its range is $[-2^m, 2^m - 2^{-n}]$*
- *its resolution is 2^{-n}*

For the unsigned integer format used in the MMDVSQ's square root calculation, an u(nsigned)Qm.n notation requires m+n bits (m+n = 32) for the input radicand. An uQm.n format produces an uQ(m/2).(n/2) square root. As examples, consider the following tables involving the square root of 2 and square root of "pi" calculations. As expected, as the number of fractional bits (n) increases, the error between the calculated square root and the "actual" result decreases.

Table 19-2. Square Root of 2 Calculations (√2 = 1.4142135623)

RCND [Hex]	RCND Q format	Results [Hex]	RES Q Format	Decimal	% Error
0x0000_0002	uQ32.00	0x0000_0001	uQ16.00	1.0	-29.289%
0x0002_0000	uQ16.16	0x0000_016A	uQ08.08	1.4140625	-0.011%
0x0200_0000	uQ08.24	0x0000_16A0	uQ04.12	1.4140625	-0.011%
0x2000_0000	uQ04.28	0x0000_5A82	uQ02.14	1.4141845703	-0.002%
0x8000_0000	uQ02.30	0x0000_B504	uQ01.15	1.4141845703	-0.002%

Table 19-3. Square Root of Pi Calculations (√Pi = 1.7724538509)

RCND [Hex]	RCND Q format	Results [Hex]	RES Q Format	Decimal	% Error
0x0000_0003	uQ32.0	0x0000_0001	uQ16.00	1.0	-43.581%
0x0003_243F	uQ16.16	0x0000_01C5	uQ08.08	1.76953125	-0.165%
0x0324_3F6A	uQ08.24	0x0000_1C5B	uQ04.12	1.772216769	-0.013%
0x3243_F6A8	uQ04.28	0x0000_716F	uQ02.14	1.7723999023	-0.003%
0xC90F_DAA0	uQ02.30	0x0000_E2DF	uQ01.15	1.7724304199	-0.001%

The application of the Q notation for square root calculations provides a powerful extension for these types of fractional numeric computations using fixed-point integer processing hardware.

19.4.2 Execution times

The MMDVSQ module includes early termination logic to finish both divide and square root calculations as quickly as possible, based on the magnitude of the input operand. Accordingly, the execution time for the calculations is data dependent as defined in Table 19-4 and Table 19-5. In this context, the execution time is defined from the register write to initiate the calculation until the result register has been updated and available to read. Stated differently, it represents the time CSR[BUSY] is asserted for a given calculation. In the following two tables, "x" signals a bit with a don't care value.

Table 19-4. Divide Execution Times

CSR[USGN] ? DEND[31:0] // unsigned divide : abs(DEND[31:0]) // signed divide	Execution Time with CSR[BUSY] = 1 [cycles]
(01,1x)xx_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	17
00(01,1x)_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	16
0000_(01,1x)xx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	15
0000_00(01,1x)_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	14
0000_0000_(01,1x)xx_xxxx__xxxx_xxxx_xxxx_xxxx	13
0000_0000_00(01,1x)_xxxx__xxxx_xxxx_xxxx_xxxx	12
0000_0000_0000_(01,1x)xx__xxxx_xxxx_xxxx_xxxx	11
0000_0000_0000_00(01,1x)__xxxx_xxxx_xxxx_xxxx	10
0000_0000_0000_0000__(01,1x)xx_xxxx_xxxx_xxxx	9
0000_0000_0000_0000__00(01,1x)_xxxx_xxxx_xxxx	8
0000_0000_0000_0000__0000_(01,1x)xx_xxxx_xxxx	7
0000_0000_0000_0000__0000_00(01,1x)_xxxx_xxxx	6
0000_0000_0000_0000__0000_0000_(01,1x)xx_xxxx	5
0000_0000_0000_0000__0000_0000_00(01,1x)_xxxx	4
0000_0000_0000_0000__0000_0000_0000_(01,1x)xx	3
0000_0000_0000_0000__0000_0000_0000_00(01,1x)	2
0000_0000_0000_0000__0000_0000_0000_0000	1

Table 19-5. Square Root Execution Times

RCND[31:0]	Execution Time with CSR[BUSY] = 1 [cycles]
(01,1x)xx_xxxx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	17
00(01,1x)_xxxx_xxxx_xxxx_x_xxx_xxxx_xxxx_xxxx	16
0000_(01,1x)xx_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	15
0000_00(01,1x)_xxxx_xxxx__xxxx_xxxx_xxxx_xxxx	14
0000_0000_(01,1x)xx_xxxx__xxxx_xxxx_xxxx_xxxx	13
0000_0000_00(01,1x)_xxxx__xxxx_xxxx_xxxx_xxxx	12
0000_0000_0000_(01,1x)xx__xxxx_xxxx_xxxx_xxxx	11
0000_0000_0000_00(01,1x)__xxxx_xxxx_xxxx_xxxx	10
0000_0000_0000_0000__(01,1x)xx_xxxx_xxxx_xxxx	9
0000_0000_0000_0000__00(01,1x)_xxxx_xxxx_xxxx	8
0000_0000_0000_0000__0000_(01,1x)xx_xxxx_xxxx	7
0000_0000_0000_0000__0000_00(01,1x)_xxxx_xxxx	6
0000_0000_0000_0000__0000_0000_(01,1x)xx_xxxx	5
0000_0000_0000_0000__0000_0000_00(01,1x)_xxxx	4
0000_0000_0000_0000__0000_0000_0000_(01,1x)xx	3
0000_0000_0000_0000__0000_0000_0000_00(01,1x)	2
0000_0000_0000_0000__0000_0000_0000_0000	2

19.4.3 Software interface

The programming model of the MMDVSQ is organized to be similar to the input arguments passed to software libraries for integer divide and square root functions.

19.4.3.1 Operation activation and result retrieval

The MMDVSQ supports 2 mechanisms for initiating a divide operation:

- The default mechanism is a "fast start" where a write to the DSOR register begins the divide.
- Alternatively, the start mechanism can begin after a write to the CSR register with the CSR[SRT] set.

The CSR[DFS] indicator selects the divide start mechanism.

```
if CSR[DFS] = 0
   then a divide is initiated by a write to the DSOR register
   else a divide is initiated by a write to the CSR register with CSR[SRT] = 1
```

A square root calculation is initiated by a write to the RCND register.

For both divide and square root calculations, the result of the operation is retrieved by reading the RES register. A memory read of this register while the calculation is still being performed causes the access to be stalled via the insertion of bus wait states until the new result is loaded into the register. Note a stalled bus cycle cannot be interrupted, so if system interrupt latency is a concern, the processor should execute a simple wait loop, for example, polling CSR[BUSY], before reading the RES register. This code construct is fully interruptible, so interrupt latency is minimized.

19.4.3.2 Context save and restore

Given that multiple memory-mapped register accesses are needed for each divide and square root calculation, interrupts may occur during the required sequence of operations. As a result, the MMDVSQ's programming model can be saved at entry to an interrupt service routine (ISR) and then restored when redispatching to the interrupted task.

The module's context can be saved by reading the DEND, DSOR, CSR, and RES registers and storing them as part of the task state. There is one special consideration for the task state save. If the last calculation was a zero divide and the divide-by-zero enable is set (CSR[DZE] = 1), then a read of the RES register is error terminated. To avoid a zero-divide error termination during a context save, the following sequence can be used:

1. Read DEND, DSOR, and CSR registers and save the values as part of the task state.

2. Clear CSR[DZE].
3. Read the RES register and save its value as part of the task state.

When restoring the context, special care must be taken to not initiate another divide calculation. Specifically, CSR[DFS] must be set first before reloading the DEND and DSOR registers. For example, the following sequence can be used for the context reload:

1. Write 0x0000_0020 to the CSR to disable the fast start mechanism.
2. Reload DEND, DSOR, CSR, and RES registers from the saved state.

Since the original context save of the control/status register is guaranteed to have CSR[SRT] = 0, there is no divide operation initiated when this register is reloaded in step 2.

Chapter 20
Miscellaneous Control Module (MCM)

20.1 Introduction

NOTE

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

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

20.1.1 Features

The MCM includes the following features:

- Program-visible information on the platform configuration
- Flash controller speculation buffer and cache configurations

20.2 Memory map/register descriptions

The memory map and register descriptions found here describe the registers using byte addresses. The registers can be written only when in supervisor mode.

MCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_3008	Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)	16	R	0007h	20.2.1/388
F000_300A	Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)	16	R	0005h	20.2.2/388

Table continues on the next page...

MCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_300C	Platform Control Register (MCM_PLACR)	32	R/W	0000_0000h	20.2.3/389
F000_3040	Compute Operation Control Register (MCM_CPO)	32	R/W	0000_0000h	20.2.4/392

20.2.1 Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)

PLASC is a 16-bit read-only register identifying the presence/absence of bus slave connections to the device's crossbar switch.

Address: F000_3000h base + 8h offset = F000_3008h

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

MCM_PLASC field descriptions

Field	Description
15–8 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 whether there is a corresponding connection to the crossbar switch's slave input port. 0 A bus slave connection to AXBS input port *n* is absent. 1 A bus slave connection to AXBS input port *n* is present.

20.2.2 Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)

PLAMC is a 16-bit read-only register identifying the presence/absence of bus master connections to the device's crossbar switch.

Address: F000_3000h base + Ah offset = F000_300Ah

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	0	1	0	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 whether there is a corresponding connection to the AXBS master input port. 0 A bus master connection to AXBS input port *n* is absent 1 A bus master connection to AXBS input port *n* is present

20.2.3 Platform Control Register (MCM_PLACR)

The PLACR register selects the arbitration policy for the crossbar masters and configures the flash memory controller.

The speculation buffer and cache in the flash memory controller is configurable via PLACR[15:10].

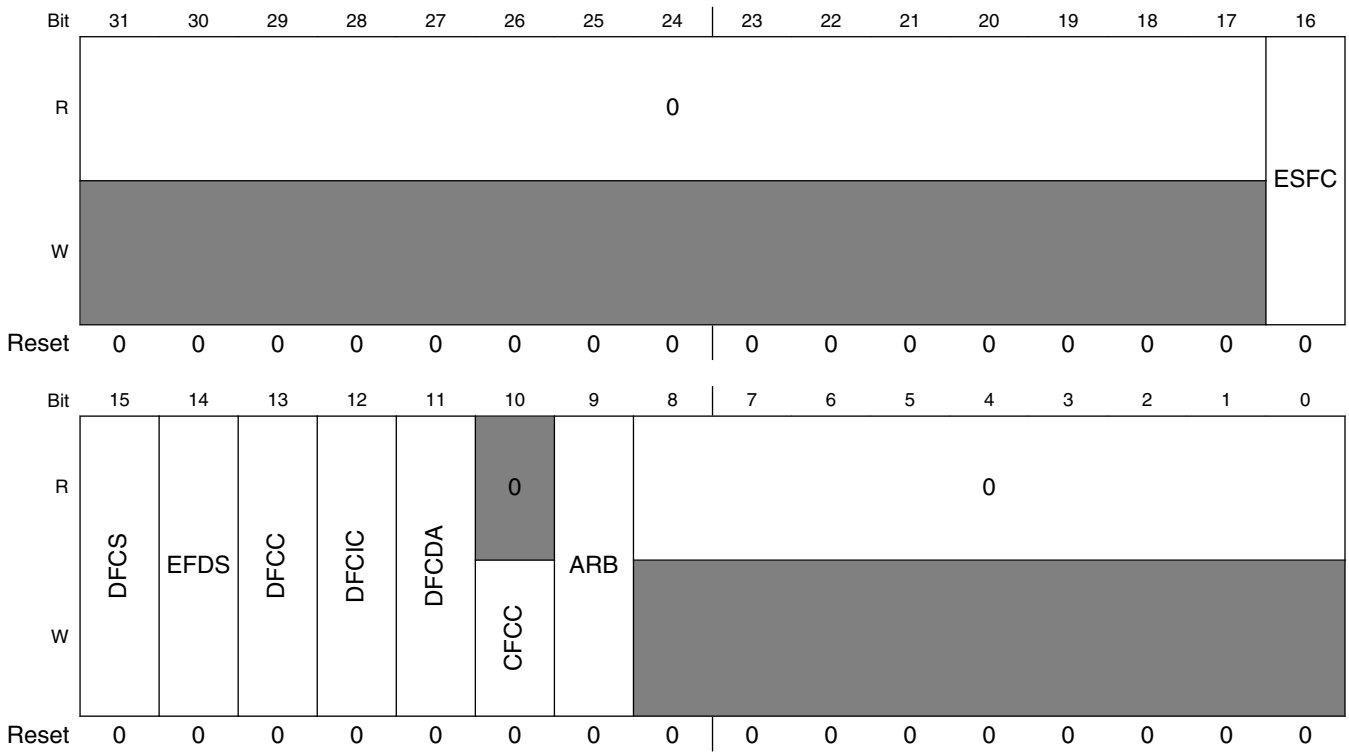
The speculation buffer is enabled only for instructions after reset. It is possible to have these states for the speculation buffer:

DFCS	EFDS	Description
0	0	Speculation buffer is on for instruction and off for data.
0	1	Speculation buffer is on for instruction and on for data.
1	X	Speculation buffer is off.

The cache in flash controller is enabled and caching both instruction and data type fetches after reset. It is possible to have these states for the cache:

DFCC	DFCIC	DFCDA	Description
0	0	0	Cache is on for both instruction and data.
0	0	1	Cache is on for instruction and off for data.
0	1	0	Cache is off for instruction and on for data.
0	1	1	Cache is off for both instruction and data.
1	X	X	Cache is off.

Address: F000_3000h base + Ch offset = F000_300Ch



MCM_PLACR field descriptions

Field	Description
31–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ESFC	Enable Stalling Flash Controller Enables stalling flash controller when flash is busy. When software needs to access the flash memory while a flash memory resource is being manipulated by a flash command, software can enable a stall mechanism to avoid a read collision. The stall mechanism allows software to execute code from the same block on which flash operations are being performed. However, software must ensure the sector the flash operations are being performed on is not the same sector from which the code is executing. ESFC enables the stall mechanism. This bit must be set only just before the flash operation is executed and must be cleared when the operation completes. 0 Disable stalling flash controller when flash is busy. 1 Enable stalling flash controller when flash is busy.
15 DFCS	Disable Flash Controller Speculation Disables flash controller speculation. 0 Enable flash controller speculation. 1 Disable flash controller speculation.
14 EFDS	Enable Flash Data Speculation Enables flash data speculation.

Table continues on the next page...

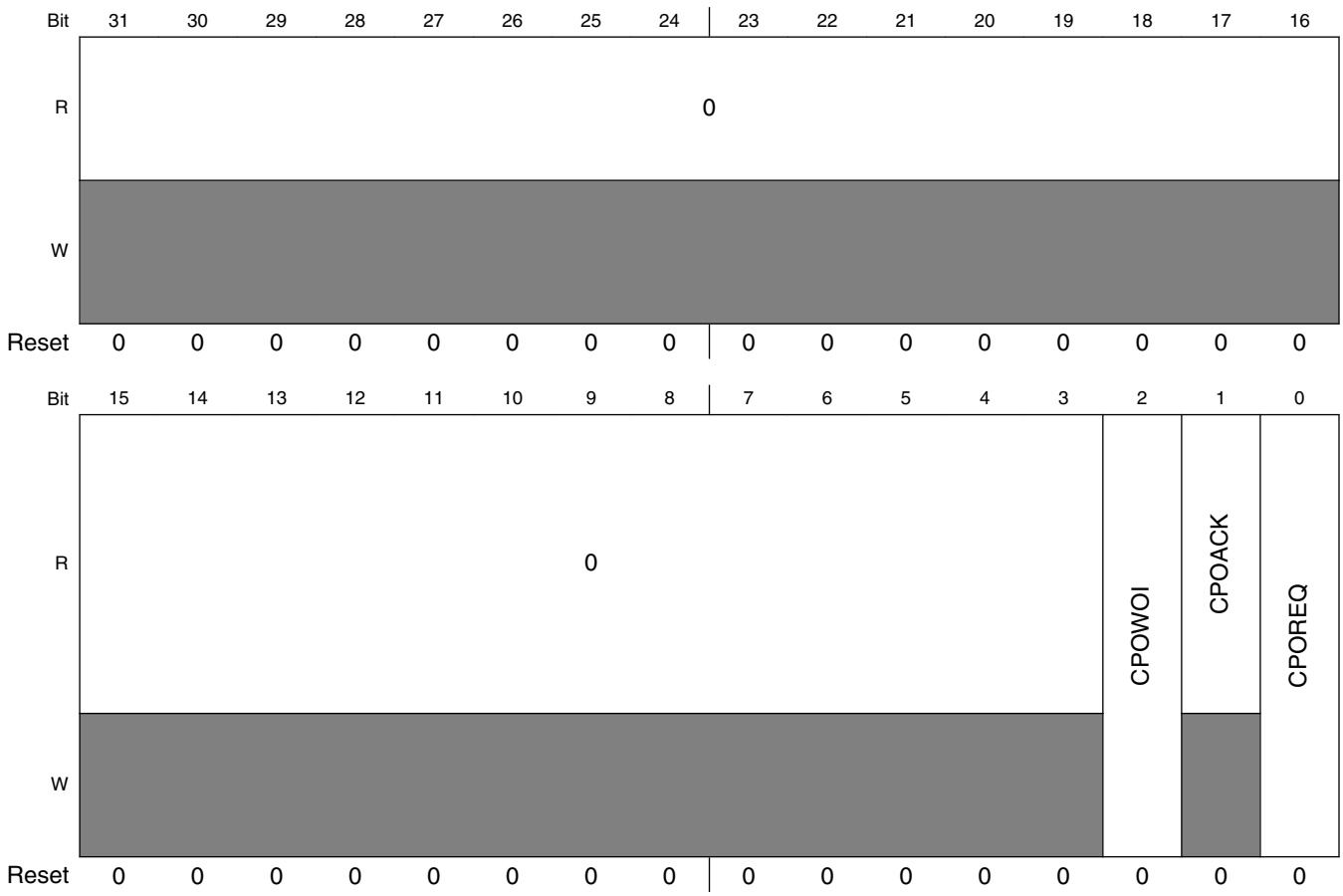
MCM_PLACR field descriptions (continued)

Field	Description
	0 Disable flash data speculation. 1 Enable flash data speculation.
13 DFCC	Disable Flash Controller Cache Disables flash controller cache. 0 Enable flash controller cache. 1 Disable flash controller cache.
12 DFCIC	Disable Flash Controller Instruction Caching Disables flash controller instruction caching. 0 Enable flash controller instruction caching. 1 Disable flash controller instruction caching.
11 DFCDA	Disable Flash Controller Data Caching Disables flash controller data caching. 0 Enable flash controller data caching 1 Disable flash controller data caching.
10 CFCC	Clear Flash Controller Cache Writing a 1 to this field clears the cache. Writing a 0 to this field is ignored. This field always reads as 0.
9 ARB	Arbitration select 0 Fixed-priority arbitration for the crossbar masters 1 Round-robin arbitration for the crossbar masters
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

20.2.4 Compute Operation Control Register (MCM_CPO)

This register controls the Compute Operation.

Address: F000_3000h base + 40h offset = F000_3040h



MCM_CPO field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CPOWOI	Compute Operation Wake-up on Interrupt 0 No effect. 1 When set, the CPOREQ is cleared on any interrupt or exception vector fetch.
1 CPOACK	Compute Operation Acknowledge 0 Compute operation entry has not completed or compute operation exit has completed. 1 Compute operation entry has completed or compute operation exit has not completed.
0 CPOREQ	Compute Operation Request This bit is auto-cleared by vector fetching if CPOWOI = 1.

Table continues on the next page...

MCM_CPO field descriptions (continued)

Field	Description
	0 Request is cleared. 1 Request Compute Operation.

Chapter 21
Micro Trace Buffer (MTB)

21.1 Introduction

Microcontrollers using the Cortex-M0+ processor core include support for a CoreSight Micro Trace Buffer to provide program trace capabilities.

The proper name for this function is the CoreSight Micro Trace Buffer for the Cortex-M0+ Processor; in this document, it is simply abbreviated as the MTB.

The simple program trace function creates instruction address change-of-flow data packets in a user-defined region of the system RAM. Accordingly, the system RAM controller manages requests from two sources:

- AMBA-AHB reads and writes from the system bus
- program trace packet writes from the processor

As part of the MTB functionality, there is a DWT (Data Watchpoint and Trace) module that allows the user to define watchpoint addresses, or optionally, an address and data value, that when triggered, can be used to start or stop the program trace recording.

This document details the functionality of both the MTB_RAM and MTB_DWT capabilities.

21.1.1 Overview

A generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers is shown as follows:

As shown in the block diagram, the platform RAM (PRAM) controller connects to two input buses:

- the crossbar slave port for system bus accesses
- a "private execution MTB port" from the core

The logical paths from the crossbar master input ports to the PRAM controller are highlighted along with the private execution trace port from the processor core. The private MTB port signals the instruction address information needed for the 64-bit program trace packets written into the system RAM. The PRAM controller output interfaces to the attached RAM array. In this document, the PRAM controller is the MTB_RAM controller.

The following information is taken from the ARM CoreSight Micro Trace Buffer documentation.

"The execution trace packet consists of a pair of 32-bit words that the MTB generates when it detects the processor PC value changes non-sequentially. A non-sequential PC change can occur during branch instructions or during exception entry.

The processor can cause a trace packet to be generated for any instruction.

The following figure shows how the execution trace information is stored in memory as a sequence of packets.

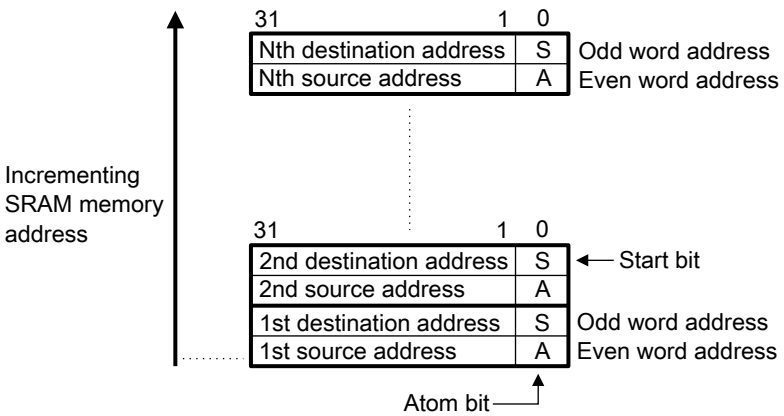


Figure 21-1. MTB execution trace storage format

The first, lower addressed, word contains the source of the branch, the address it branched from. The value stored only records bits[31:1] of the source address, because Thumb instructions are at least halfword aligned. The least significant bit of the value is the A-bit. The A-bit indicates the atomic state of the processor at the time of the branch, and can differentiate whether the branch originated from an instruction in a program, an exception, or a PC update in Debug state. When it is zero the branch originated from an instruction, when it is one the branch originated from an exception or PC update in Debug state. This word is always stored at an even word location.

The second, higher addressed word contains the destination of the branch, the address it branched to. The value stored only records bits[31:1] of the branch address. The least significant bit of the value is the S-bit. The S-bit indicates where the trace started. An S-bit value of 1 indicates where the first packet after the trace started and a value of 0 is

used for other packets. Because it is possible to start and stop tracing multiple times in a trace session, the memory might contain several packets with the S-bit set to 1. This word is always stored in the next higher word in memory, an odd word address.

When the A-bit is set to 1, the source address field contains the architecturally-preferred return address for the exception. For example, if an exception was caused by an SVC instruction, then the source address field contains the address of the following instruction. This is different from the case where the A-bit is set to 0. In this case, the source address contains the address of the branch instruction.

For an exception return operation, two packets are generated:

- The first packet has the:
  - Source address field set to the address of the instruction that causes the exception return, BX or POP.
  - Destination address field set to bits[31:1] of the EXC_RETURN value. See the ARM v6-M Architecture Reference Manual.
  - The A-bit set to 0.
- The second packet has the:
  - Source address field set to bits[31:1] of the EXC_RETURN value.
  - Destination address field set to the address of the instruction where execution commences.
  - A-bit set to 1."

Given the recorded change-of-flow trace packets in system RAM and the memory image of the application, a debugger can read out the data and create an instruction-by-instruction program trace. In keeping with the low area and power implementation cost design targets, the MTB trace format is less efficient than other CoreSight trace modules, for example, the ETM (Embedded Trace Macrocell). Since each branch packet is 8 bytes in size, a 1 KB block of system RAM can contain 128 branches. Using the Dhrystone 2.1 benchmark's dynamic runtime as an example, this corresponds to about 875 instructions per KB of trace RAM, or with a zero wait state memory, this corresponds to approximately 1600 processor cycles per KB. This metric is obviously very sensitive to the runtime characteristics of the user code.

The MTB_DWT function (not shown in the core platform block diagram) monitors the processor address and data buses so that configurable watchpoints can be detected to trigger the appropriate response in the MTB recording.

21.1.2 Features

The key features of the MTB_RAM and MTB_DWT include:

- Memory controller for system RAM and Micro Trace Buffer for program trace packets
- Read/write capabilities for system RAM accesses, write-only for program trace packets
- Supports zero wait state response to system bus accesses when no trace data is being written
- Can buffer two AHB address phases and one data write for system RAM accesses
- Supports 64-bit program trace packets including source and destination instruction addresses
- Program trace information in RAM available to MCU's application code or external debugger
- Program trace watchpoint configuration accessible by MCU's application code or debugger
- Location and size of RAM trace buffer is configured by software
- Two DWT comparators (addresses or address + data) provide programmable start/stop recording
- CoreSight compliant debug functionality

21.1.3 Modes of operation

The MTB_RAM and MTB_DWT functions do not support any special modes of operation. The MTB_RAM controller, as a memory-mapped device located on the platform's slave AHB system bus, responds strictly on the basis of memory addresses for accesses to its attached RAM array. The MTB private execution bus provides program trace packet write information to the RAM controller. Both the MTB_RAM and MTB_DWT modules are memory-mapped, so their programming models can be accessed.

All functionality associated with the MTB_RAM and MTB_DWT modules resides in the core platform's clock domain; this includes its connections with the RAM array.

21.2 External signal description

The MTB_RAM and MTB_DWT modules do not directly support any external interfaces.

The internal interface includes a standard AHB bus with a 32-bit datapath width from the appropriate crossbar slave port plus the private execution trace bus from the processor core. The signals in the private execution trace bus are detailed in the following table taken from the ARM CoreSight Micro Trace Buffer documentation. The signal direction is defined as viewed by the MTB_RAM controller.

Table 21-1. Private execution trace port from the core to MTB_RAM

Signal	Direction	Description
LOCKUP	Input	Indicates the processor is in the Lockup state. This signal is driven LOW for cycles when the processor is executing normally and driven HIGH for every cycle the processor is waiting in the Lockup state. This signal is valid on every cycle.
IAESEQ	Input	Indicates the next instruction address in execute, IAEX, is sequential, that is non-branching.
IAEXEN	Input	IAEX register enable.
IAEX[30:0]	Input	Registered address of the instruction in the execution stage, shifted right by one bit, that is, PC >> 1.
ATOMIC	Input	Indicates the processor is performing non-instruction related activities.
EDBGRQ	Output	Request for the processor to enter the Debug state, if enabled, and halt.

In addition, there are two signals formed by the MTB_DWT module and driven to the MTB_RAM controller: TSTART (trace start) and TSTOP (trace stop). These signals can be configured using the trace watchpoints to define programmable addresses and data values to affect the program trace recording state.

21.3 Memory map and register definition

The MTB_RAM and MTB_DWT modules each support a sparsely-populated 4 KB address space for their programming models. For each address space, there are a variety of control and configurable registers near the base address, followed by a large unused address space and finally a set of CoreSight registers to support dynamic determination of the debug configuration for the device.

Accesses to the programming model follow standard ARM conventions. Taken from the ARM CoreSight Micro Trace Buffer documentation, these are:

- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in UNPREDICTABLE behavior.
- The behavior of the MTB is UNPREDICTABLE if the registers with UNKNOWN reset values are not programmed prior to enabling trace.
- Unless otherwise stated in the accompanying text:
  - Do not modify reserved register bits
  - Ignore reserved register bits on reads

- All register bits are reset to a logic 0 by a system or power-on reset
- Use only word size, 32-bit, transactions to access all registers

21.3.1 MTB_RAM Memory Map

MTB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_0000	MTB Position Register (MTB_POSITION)	32	R/W	Undefined	21.3.1.1/ 401
F000_0004	MTB Master Register (MTB_MASTER)	32	R/W	See section	21.3.1.2/ 403
F000_0008	MTB Flow Register (MTB_FLOW)	32	R/W	Undefined	21.3.1.3/ 404
F000_000C	MTB Base Register (MTB_BASE)	32	R	Undefined	21.3.1.4/ 406
F000_0F00	Integration Mode Control Register (MTB_MODECTRL)	32	R	0000_0000h	21.3.1.5/ 407
F000_0FA0	Claim TAG Set Register (MTB_TAGSET)	32	R	0000_0000h	21.3.1.6/ 407
F000_0FA4	Claim TAG Clear Register (MTB_TAGCLEAR)	32	R	0000_0000h	21.3.1.7/ 408
F000_0FB0	Lock Access Register (MTB_LOCKACCESS)	32	R	0000_0000h	21.3.1.8/ 408
F000_0FB4	Lock Status Register (MTB_LOCKSTAT)	32	R	0000_0000h	21.3.1.9/ 409
F000_0FB8	Authentication Status Register (MTB_AUTHSTAT)	32	R	0000_0000h	21.3.1.10/ 409
F000_0FBC	Device Architecture Register (MTB_DEVICEARCH)	32	R	4770_0A31h	21.3.1.11/ 410
F000_0FC8	Device Configuration Register (MTB_DEVICECFG)	32	R	0000_0000h	21.3.1.12/ 410
F000_0FCC	Device Type Identifier Register (MTB_DEVICETYPID)	32	R	0000_0031h	21.3.1.13/ 411
F000_0FD0	Peripheral ID Register (MTB_PERIPHID4)	32	R	See section	21.3.1.14/ 411
F000_0FD4	Peripheral ID Register (MTB_PERIPHID5)	32	R	See section	21.3.1.14/ 411
F000_0FD8	Peripheral ID Register (MTB_PERIPHID6)	32	R	See section	21.3.1.14/ 411
F000_0FDC	Peripheral ID Register (MTB_PERIPHID7)	32	R	See section	21.3.1.14/ 411
F000_0FE0	Peripheral ID Register (MTB_PERIPHID0)	32	R	See section	21.3.1.14/ 411

Table continues on the next page...

MTB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_0FE4	Peripheral ID Register (MTB_PERIPHID1)	32	R	See section	21.3.1.14/ 411
F000_0FE8	Peripheral ID Register (MTB_PERIPHID2)	32	R	See section	21.3.1.14/ 411
F000_0FEC	Peripheral ID Register (MTB_PERIPHID3)	32	R	See section	21.3.1.14/ 411
F000_0FF0	Component ID Register (MTB_COMPID0)	32	R	See section	21.3.1.15/ 412
F000_0FF4	Component ID Register (MTB_COMPID1)	32	R	See section	21.3.1.15/ 412
F000_0FF8	Component ID Register (MTB_COMPID2)	32	R	See section	21.3.1.15/ 412
F000_0FFC	Component ID Register (MTB_COMPID3)	32	R	See section	21.3.1.15/ 412

21.3.1.1 MTB Position Register (MTB_POSITION)

The MTB_POSITION register contains the Trace Write Address Pointer and Wrap fields. This register can be modified by the explicit programming model writes. It is also automatically updated by the MTB hardware when trace packets are being recorded.

The base address of the system RAM in the memory map dictates special consideration for the placement of the MTB. Consider the following guidelines:

For the standard configuration where the size of the MTB is ≤ 25% of the total RAM capacity, it is recommended the MTB be based at the address defined by the MTB_BASE register. The read-only MTB_BASE register is defined by the expression (0x2000_0000 - (RAM_Size/4)). For this configuration, the MTB_POSITION register is initialized to MTB_BASE & 0x0000_7FF8.

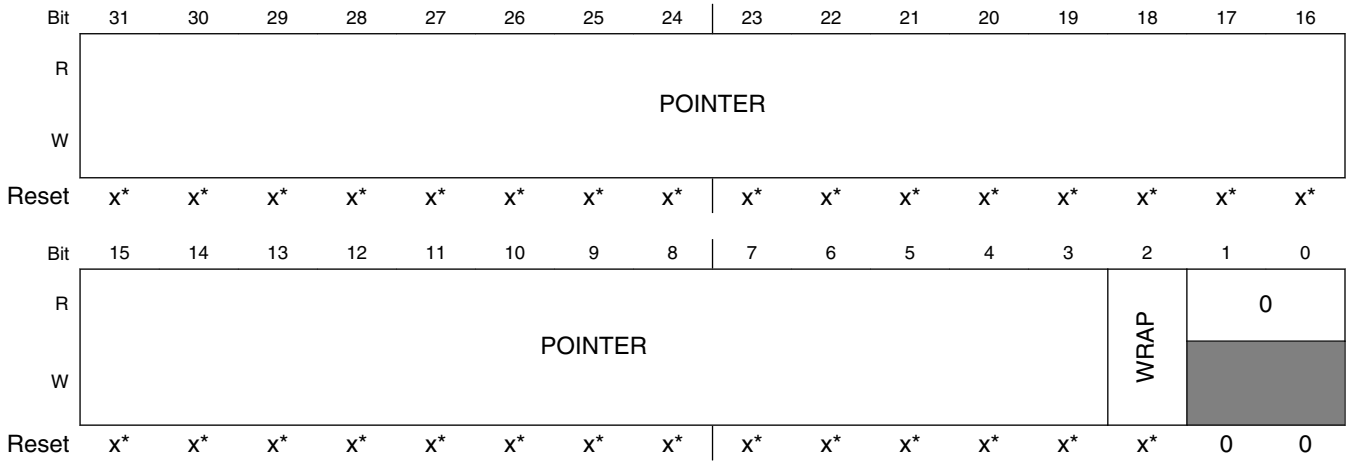
If the size of the MTB is more than 25% but less than or equal to 50% of the total RAM capacity, it is recommended the MTB be based at address 0x2000_0000. In this configuration, the MTB_POSITION register is initialized to (0x2000_0000 & 0x0000_7FF8) = 0x0000_00000.

Following these two suggested placements provides a full-featured circular memory buffer containing program trace packets.

In the unlikely event an even larger trace buffer is required, a write-once capacity of 75% of the total RAM capacity can be based at address 0x2000_0000. The MTB_POSITION register is initialized to (0x2000_0000 & 0x0000_7FF8) = 0x0000_0000. However, this configuration cannot support operation as a circular queue and instead requires the use of

the MTB_FLOW[WATERMARK] capability to automatically disable tracing or halting the processor as the number of packet writes approach the buffer capacity. See the MTB_FLOW register description for more details.

Address: F000_0000h base + 0h offset = F000_0000h



* Notes:
- x = Undefined at reset.

MTB_POSITION field descriptions

Field	Description

31–3
POINTER
	Trace Packet Address Pointer[28:0]
	Because a packet consists of two words, the POINTER field is the address of the first word of a packet. This field contains bits[31:3] of the RAM address where the next trace packet is written. Therefore, it points to an unused location and is automatically incremented.
	A debug agent can calculate the system memory map address for the current location in the MTB using the following "generic" equation:
	Given mtb_size = 1 << (MTB_MASTER[MASK] + 4),
	systemAddress = MTB_BASE + (((MTB_POSITION & 0xFFFF_FFF8) + (mtb_size - (MTB_BASE & (mtb_size-1)))) & 0x0000_7FF8);
	For this device, a simpler expression also applies. See the following pseudo-code:
	if ((MTB_POSITION >> 13) == 0x3) systemAddress = (0x1FFF << 16) + (0x1 << 15) + (MTB_POSITION & 0x7FF8); else systemAddress = (0x2000 << 16) + (0x0 << 15) + (MTB_POSITION & 0x7FF8);
	NOTE: The size of the RAM is parameterized and the most significant bits of the POINTER field are RAZ/WI.
	For these devices, POSITION[31:15] == POSITION[POINTER[28:12]] are RAZ/WI. Therefore, the active bits in this field are POSITION[14:3] == POSITION[POINTER[11:0]].

2 WRAP	WRAP This field is set to 1 automatically when the POINTER value wraps as determined by the MTB_MASTER[MASK] field in the MASTER Trace Control Register. A debug agent might use the WRAP field to determine whether the trace information above and below the pointer address is valid.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.3.1.2 MTB Master Register (MTB_MASTER)

The MTB_MASTER register contains the main program trace enable plus other trace controls. This register can be modified by the explicit programming model writes. MTB_MASTER[EN] and MTB_MASTER[HALTREQ] fields are also automatically updated by the MTB hardware.

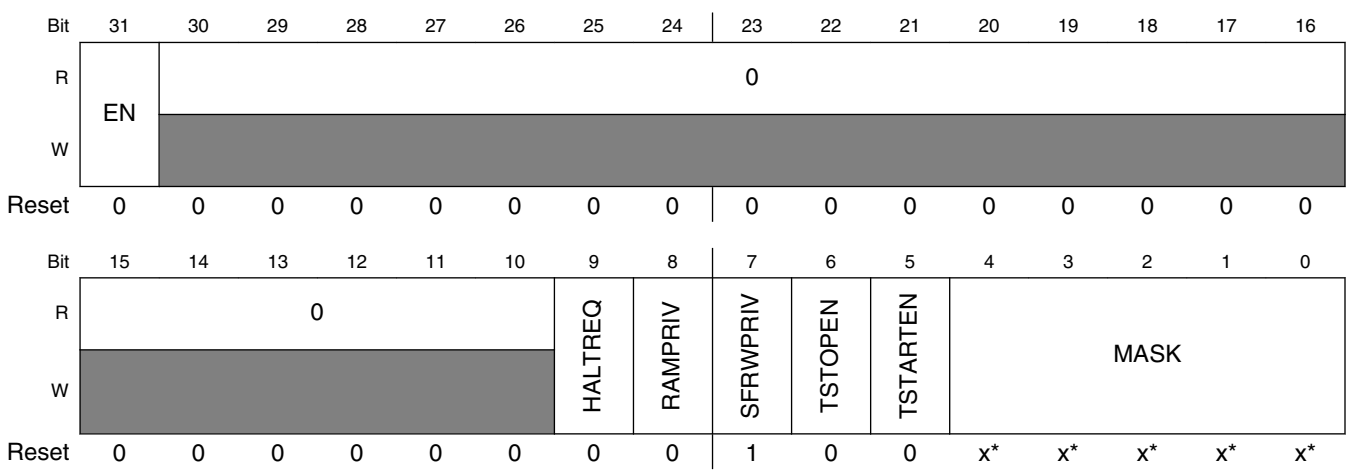
Before MTB_MASTER[EN] or MTB_MASTER[TSTARTEN] are set to 1, the software must initialize the MTB_POSITION and MTB_FLOW registers.

If MTB_FLOW[WATERMARK] is used to stop tracing or to halt the processor, MTB_MASTER[MASK] must still be set to a value that prevents MTB_POSITION[POINTER] from wrapping before it reaches the MTB_FLOW[WATERMARK] value.

NOTE

The format of this mask field is different than MTBDWT_MASKn[MASK].

Address: F000_0000h base + 4h offset = F000_0004h



* Notes:
- x = Undefined at reset.

MTB_MASTER field descriptions

Field	Description
31 EN	Main Trace Enable When this field is 1, trace data is written into the RAM memory location addressed by MTB_POSITION[POINTER]. The MTB_POSITION[POINTER] value auto increments after the trace data packet is written. EN can be automatically set to 0 using the MTB_FLOW[WATERMARK] field and the MTB_FLOW[AUTOSTOP] bit.

Table continues on the next page...

MTB_MASTER field descriptions (continued)

Field	Description
	EN is automatically set to 1 if TSTARTEN is 1 and the TSTART signal is HIGH. EN is automatically set to 0 if TSTOPEN is 1 and the TSTOP signal is HIGH. **NOTE:** If EN is set to 0 because MTB_FLOW[WATERMARK] is set, then it is not automatically set to 1 if TSTARTEN is 1 and the TSTART input is HIGH. In this case, tracing can only be restarted if MTB_FLOW[WATERMARK] or MTB_POSITION[POINTER] value is changed by software.
30–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 HALTREQ	Halt Request This field is connected to the halt request signal of the trace logic, EDBGRQ. When HALTREQ is set to 1, the EDBFGRQ is asserted if DBGEN (invasive debug enable, one of the debug authentication interface signals) is also HIGH. HALTREQ can be automatically set to 1 using MTB_FLOW[WATERMARK].
8 RAMPRIV	RAM Privilege If this field is 0, then user or privileged AHB read and write accesses to the RAM are permitted. If this field is 1, then only privileged AHB read and write accesses to the RAM are permitted and user accesses are RAZ/WI. The HPROT[1] signal determines if an access is a user or privileged mode reference.
7 SFRWPRIV	Special Function Register Write Privilege If this field is 0, then user or privileged AHB read and write accesses to the MTB_RAM Special Function Registers (programming model) are permitted. If this field is 1, then only privileged write accesses are permitted; user write accesses are ignored. The HPROT[1] signal determines if an access is user or privileged. Note MTB_RAM SFR read access are not controlled by this bit and are always permitted.
6 TSTOPEN	Trace Stop Input Enable If this field is 1 and the TSTOP signal is HIGH, then EN is set to 0. If a trace packet is being written to memory, the write is completed before tracing is stopped.
5 TSTARTEN	Trace Start Input Enable If this field is 1 and the TSTART signal is HIGH, then EN is set to 1. Tracing continues until a stop condition occurs.
MASK	Mask This value determines the maximum size of the trace buffer in RAM. It specifies the most-significant bit of the MTB_POSITION[POINTER] field that can be updated by automatic increment. If the trace tries to advance past this power of 2, the MTB_POSITION[WRAP] bit is set to 1, the MTB_POSITION[MASK+3:3] == MTB_POSITION[POINTER[MASK:0]] bits are set to 0, and the MTB_POSITION[14:MASK+3] == MTB_POSITION[POINTER[11:MASK+1]] bits remain unchanged. This field causes the trace packet information to be stored in a circular buffer of size 2^[MASK+4] bytes, that can be positioned in memory at multiples of this size. As detailed in the MTB_POSITION description, typical "upper limits" for the MTB size are RAM_Size/4 or RAM_Size/2. Values greater than the maximum have the same effect as the maximum.

21.3.1.3 MTB Flow Register (MTB_FLOW)

The MTB_FLOW register contains the watermark address and the autostop/autohalt control bits.

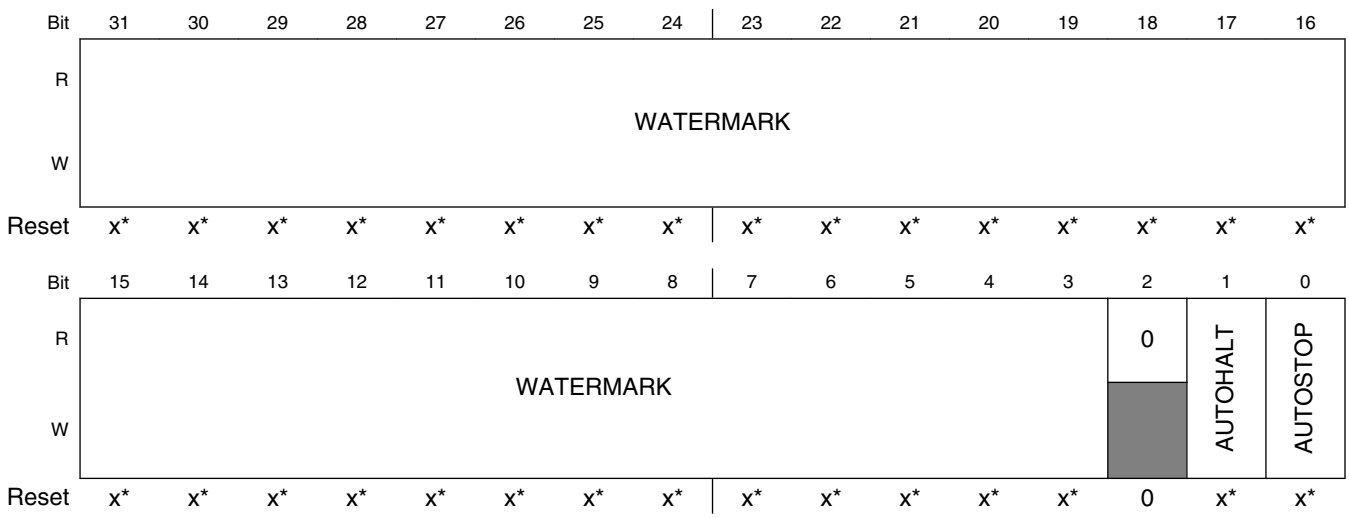
If tracing is stopped using the watermark autostop feature, it cannot be restarted until software clears the watermark autostop. This can be achieved in one of the following ways:

- Changing the MTB_POSITION[POINTER] field value to point to the beginning of the trace buffer, or
- Setting MTB_FLOW[AUTOSTOP] = 0.

A debug agent can use MTB_FLOW[AUTOSTOP] to fill the trace buffer once only without halting the processor.

A debug agent can use MTB_FLOW[AUTOHALT] to fill the trace buffer once before causing the Cortex-M0+ processor to enter the Debug state. To enter Debug state, the Cortex-M0+ processor might have to perform additional branch type operations. Therefore, the MTB_FLOW[WATERMARK] field must be set below the final entry in the trace buffer region.

Address: F000_0000h base + 8h offset = F000_0008h



* Notes:
- x = Undefined at reset.

MTB_FLOW field descriptions

Field	Description
31–3 WATERMARK	WATERMARK[28:0] This field contains an address in the same format as the MTB_POSITION[POINTER] field. When MTB_POSITION[POINTER] matches the WATERMARK field value, actions defined by the AUTOHALT and AUTOSTOP bits are performed.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MTB_FLOW field descriptions (continued)

Field	Description
1 AUTOHALT	AUTOHALT If this field is 1 and WATERMARK is equal to MTB_POSITION[POINTER], then MTB_MASTER[HALTREQ] is automatically set to 1. If the DBGEN signal is HIGH, the MTB asserts this halt request to the Cortex-M0+ processor by asserting the EDBGRQ signal.
0 AUTOSTOP	AUTOSTOP If this field is 1 and WATERMARK is equal to MTB_POSITION[POINTER], then MTB_MASTER[EN] is automatically set to 0. This stops tracing.

21.3.1.4 MTB Base Register (MTB_BASE)

The read-only MTB_BASE Register indicates where the RAM is located in the system memory map. This register is provided to enable auto discovery of the MTB RAM location, by a debug agent and is defined by a hardware design parameter. For this device, the base address is defined by the expression: MTB_BASE[BASEADDR] = 0x2000_0000 - (RAM_Size/4)

Address: F000_0000h base + Ch offset = F000_000Ch

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	BASEADDR																															
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.

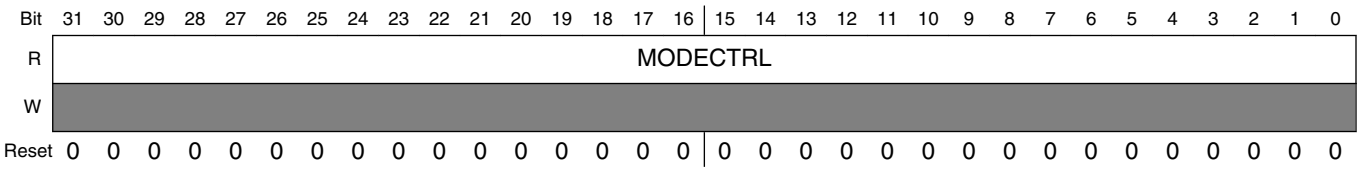
MTB_BASE field descriptions

Field	Description
BASEADDR	BASEADDR This value is defined with a hardwired signal and the expression: 0x2000_0000 - (RAM_Size/4). For example, if the total RAM capacity is 16 KB, this field is 0x1FFF_F000.

21.3.1.5 Integration Mode Control Register (MTB_MODECTRL)

This register enables the device to switch from a functional mode, or default behavior, into integration mode. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + F00h offset = F000_0F00h



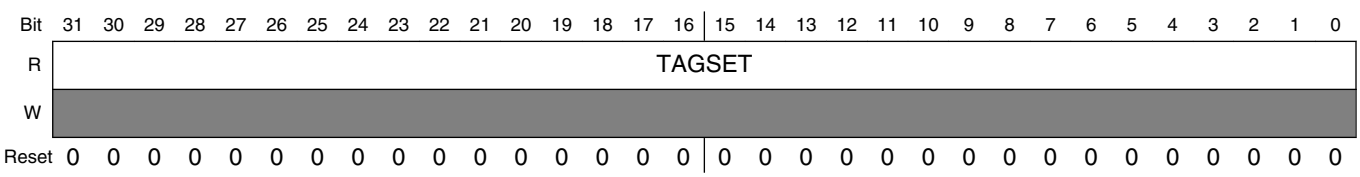
MTB_MODECTRL field descriptions

Field	Description
MODECTRL	MODECTRL Hardwired to 0x0000_0000

21.3.1.6 Claim TAG Set Register (MTB_TAGSET)

The Claim Tag Set Register returns the number of bits that can be set on a read, and enables individual bits to be set on a write. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FA0h offset = F000_0FA0h



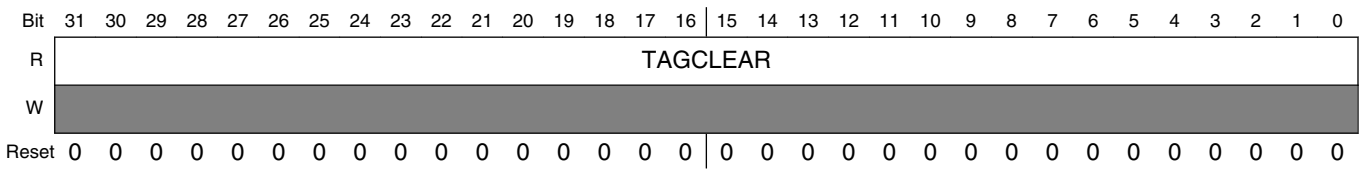
MTB_TAGSET field descriptions

Field	Description
TAGSET	TAGSET Hardwired to 0x0000_0000

21.3.1.7 Claim TAG Clear Register (MTB_TAGCLEAR)

The read/write Claim Tag Clear Register is used to read the claim status on debug resources. A read indicates the claim tag status. Writing 1 to a specific bit clears the corresponding claim tag to 0. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FA4h offset = F000_0FA4h



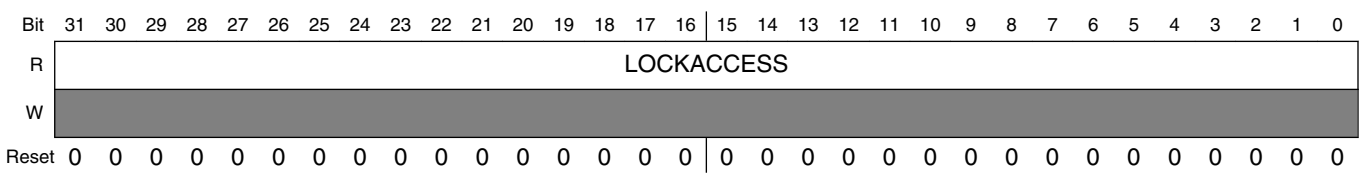
MTB_TAGCLEAR field descriptions

Field	Description
TAGCLEAR	TAGCLEAR Hardwired to 0x0000_0000

21.3.1.8 Lock Access Register (MTB_LOCKACCESS)

The Lock Access Register enables a write access to component registers. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FB0h offset = F000_0FB0h



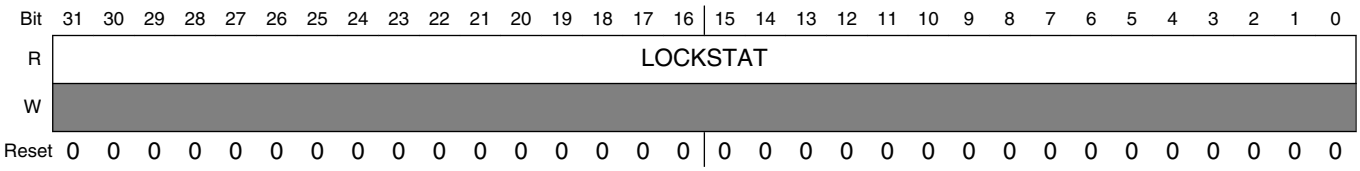
MTB_LOCKACCESS field descriptions

Field	Description
LOCKACCESS	Hardwired to 0x0000_0000

21.3.1.9 Lock Status Register (MTB_LOCKSTAT)

The Lock Status Register indicates the status of the lock control mechanism. This register is used in conjunction with the Lock Access Register. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FB4h offset = F000_0FB4h



MTB_LOCKSTAT field descriptions

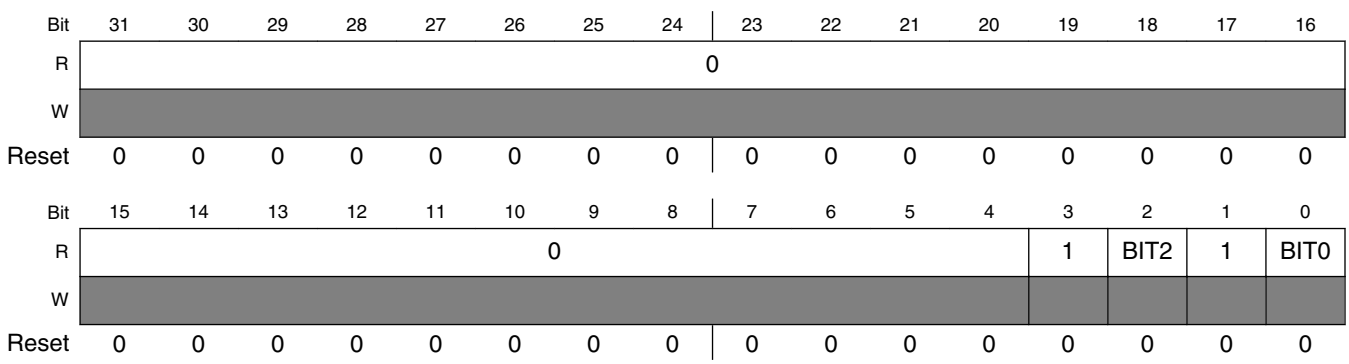
Field	Description
LOCKSTAT	LOCKSTAT Hardwired to 0x0000_0000

21.3.1.10 Authentication Status Register (MTB_AUTHSTAT)

The Authentication Status Register reports the required security level and current status of the security enable bit pairs. Where functionality changes on a given security level, this change must be reported in this register. It is connected to specific signals used during the auto-discovery process by an external debug agent.

MTB_AUTHSTAT[3:2] indicates if nonsecure, noninvasive debug is enabled or disabled, while MTB_AUTHSTAT[1:0] indicates the enabled/disabled state of nonsecure, invasive debug. For both 2-bit fields, 0b10 indicates the functionality is disabled and 0b11 indicates it is enabled.

Address: F000_0000h base + FB8h offset = F000_0FB8h



MTB_AUTHSTAT field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	BIT3 This read-only field is reserved and always has the value 1.
2 BIT2	BIT2 Connected to NIDEN or DBGEN signal.
1 Reserved	BIT1 This read-only field is reserved and always has the value 1.
0 BIT0	Connected to DBGEN.

21.3.1.11 Device Architecture Register (MTB_DEVICEARCH)

This register indicates the device architecture. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FBCh offset = F000_0FBCh

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	DEVICEARCH																															
W																																
Reset	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	1

MTB_DEVICEARCH field descriptions

Field	Description
DEVICEARCH	DEVICEARCH Hardwired to 0x4770_0A31.

21.3.1.12 Device Configuration Register (MTB_DEVICECFG)

This register indicates the device configuration. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FC8h offset = F000_0FC8h

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	DEVICECFG																															
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	0	0	0

MTB_DEVICECFG field descriptions

Field	Description
DEVICECFG	DEVICECFG Hardwired to 0x0000_0000.

21.3.1.13 Device Type Identifier Register (MTB_DEVICETYPID)

This register indicates the device type ID. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FCCh offset = F000_0FCCh

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	DEVICETYPID																															
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	1	0	0	0	1

MTB_DEVICETYPID field descriptions

Field	Description
DEVICETYPID	DEVICETYPID Hardwired to 0x0000_0031.

21.3.1.14 Peripheral ID Register (MTB_PERIPHID*n*)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FD0h 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	PERIPHID																															
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*

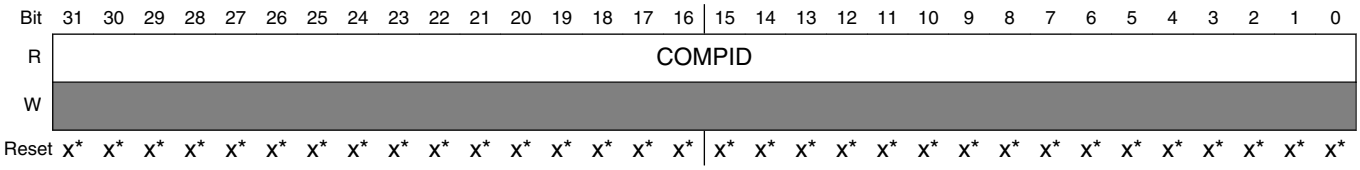
MTB_PERIPHID*n* field descriptions

Field	Description
PERIPHID	PERIPHID Peripheral ID4 is hardwired to 0x0000_0004; ID0 to 0x0000_0032; ID1 to 0x0000_00B9; ID2 to 0x0000_001B; and all the others to 0x0000_0000.

21.3.1.15 Component ID Register (MTB_COMPID*n*)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_0000h base + FF0h offset + (4d × i), where i=0d to 3d



MTB_COMPID*n* field descriptions

Field	Description
COMPID	Component ID Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0090; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.

21.3.2 MTB_DWT Memory Map

The MTB_DWT programming model supports a very simplified subset of the v7M debug architecture and follows the standard ARM DWT definition.

MTBDWT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_1000	MTB DWT Control Register (MTBDWT_CTRL)	32	R	2F00_0000h	21.3.2.1/ 413
F000_1020	MTB_DWT Comparator Register (MTBDWT_COMP0)	32	R/W	0000_0000h	21.3.2.2/ 414
F000_1024	MTB_DWT Comparator Mask Register (MTBDWT_MASK0)	32	R/W	0000_0000h	21.3.2.3/ 415
F000_1028	MTB_DWT Comparator Function Register 0 (MTBDWT_FCT0)	32	R/W	0000_0000h	21.3.2.4/ 416
F000_1030	MTB_DWT Comparator Register (MTBDWT_COMP1)	32	R/W	0000_0000h	21.3.2.2/ 414
F000_1034	MTB_DWT Comparator Mask Register (MTBDWT_MASK1)	32	R/W	0000_0000h	21.3.2.3/ 415
F000_1038	MTB_DWT Comparator Function Register 1 (MTBDWT_FCT1)	32	R/W	0000_0000h	21.3.2.5/ 418
F000_1200	MTB_DWT Trace Buffer Control Register (MTBDWT_TBCTRL)	32	R/W	2000_0000h	21.3.2.6/ 419

Table continues on the next page...

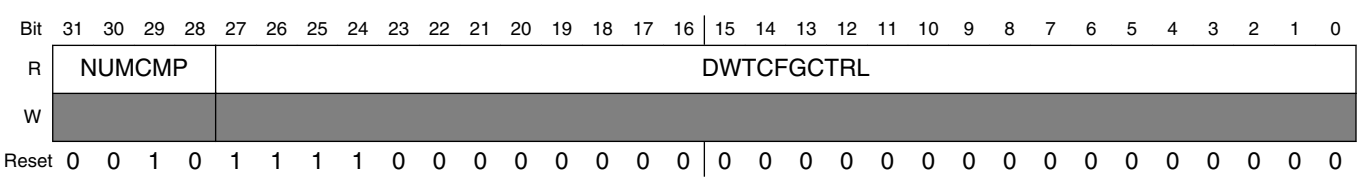
MTBDWT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_1FC8	Device Configuration Register (MTBDWT_DEVICECFG)	32	R	0000_0000h	21.3.2.7/ 421
F000_1FCC	Device Type Identifier Register (MTBDWT_DEVICETYPID)	32	R	0000_0004h	21.3.2.8/ 421
F000_1FD0	Peripheral ID Register (MTBDWT_PERIPHID4)	32	R	See section	21.3.2.9/ 422
F000_1FD4	Peripheral ID Register (MTBDWT_PERIPHID5)	32	R	See section	21.3.2.9/ 422
F000_1FD8	Peripheral ID Register (MTBDWT_PERIPHID6)	32	R	See section	21.3.2.9/ 422
F000_1FDC	Peripheral ID Register (MTBDWT_PERIPHID7)	32	R	See section	21.3.2.9/ 422
F000_1FE0	Peripheral ID Register (MTBDWT_PERIPHID0)	32	R	See section	21.3.2.9/ 422
F000_1FE4	Peripheral ID Register (MTBDWT_PERIPHID1)	32	R	See section	21.3.2.9/ 422
F000_1FE8	Peripheral ID Register (MTBDWT_PERIPHID2)	32	R	See section	21.3.2.9/ 422
F000_1FEC	Peripheral ID Register (MTBDWT_PERIPHID3)	32	R	See section	21.3.2.9/ 422
F000_1FF0	Component ID Register (MTBDWT_COMPID0)	32	R	See section	21.3.2.10/ 422
F000_1FF4	Component ID Register (MTBDWT_COMPID1)	32	R	See section	21.3.2.10/ 422
F000_1FF8	Component ID Register (MTBDWT_COMPID2)	32	R	See section	21.3.2.10/ 422
F000_1FFC	Component ID Register (MTBDWT_COMPID3)	32	R	See section	21.3.2.10/ 422

21.3.2.1 MTB DWT Control Register (MTBDWT_CTRL)

The MTBDWT_CTRL register provides read-only information on the watchpoint configuration for the MTB_DWT.

Address: F000_1000h base + 0h offset = F000_1000h



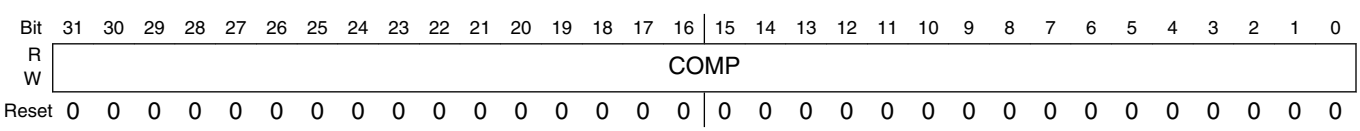
MTBDWT_CTRL field descriptions

Field	Description
31–28 NUMCMP	Number of comparators The MTB_DWT implements two comparators.
DWTCFGCTRL	DWT configuration controls This field is hardwired to 0xF00_0000, disabling all the remaining DWT functionality. The specific fields and their state are: MTBDWT_CTRL[27] = NOTRCPKT = 1, trace sample and exception trace is not supported MTBDWT_CTRL[26] = NOEXTTRIG = 1, external match signals are not supported MTBDWT_CTRL[25] = NOCYCCNT = 1, cycle counter is not supported MTBDWT_CTRL[24] = NOPRFCNT = 1, profiling counters are not supported MTBDWT_CTRL[22] = CYCEBTENA = 0, no POSTCNT underflow packets generated MTBDWT_CTRL[21] = FOLDEVTENA = 0, no folded instruction counter overflow events MTBDWT_CTRL[20] = LSUEVTENA = 0, no LSU counter overflow events MTBDWT_CTRL[19] = SLEEPEVTENA = 0, no sleep counter overflow events MTBDWT_CTRL[18] = EXCEVTENA = 0, no exception overhead counter events MTBDWT_CTRL[17] = CPIEVTENA = 0, no CPI counter overflow events MTBDWT_CTRL[16] = EXCTRCENA = 0, generation of exception trace disabled MTBDWT_CTRL[12] = PCSAMPLENA = 0, no periodic PC sample packets generated MTBDWT_CTRL[11:10] = SYNCTAP = 0, no synchronization packets MTBDWT_CTRL[9] = CYCTAP = 0, cycle counter is not supported MTBDWT_CTRL[8:5] = POSTINIT = 0, cycle counter is not supported MTBDWT_CTRL[4:1] = POSTPRESET = 0, cycle counter is not supported MTBDWT_CTRL[0] = CYCCNTENA = 0, cycle counter is not supported

21.3.2.2 MTB_DWT Comparator Register (MTBDWT_COMP*n*)

The MTBDWT_COMPn registers provide the reference value for comparator n.

Address: F000_1000h base + 20h offset + (16d × i), where i=0d to 1d



MTBDWT_COMP*n* field descriptions

Field	Description
COMP	Reference value for comparison If MTBDWT_COMP0 is used for a data value comparator and the access size is byte or halfword, the data value must be replicated across all appropriate byte lanes of this register. For example, if the data is a

MTBDWT_COMP*n* field descriptions (continued)

Field	Description
	byte-sized "x" value, then COMP[31:24] = COMP[23:16] = COMP[15:8] = COMP[7:0] = "x". Likewise, if the data is a halfword-size "y" value, then COMP[31:16] = COMP[15:0] = "y".

21.3.2.3 MTB_DWT Comparator Mask Register (MTBDWT_MASK*n*)

The MTBDWT_MASKn registers define the size of the ignore mask applied to the reference address for address range matching by comparator n. Note the format of this mask field is different than the MTB_MASTER[MASK].

Address: F000_1000h base + 24h offset + (16d × i), where i=0d to 1d

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																											MASK				
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	0	0	0

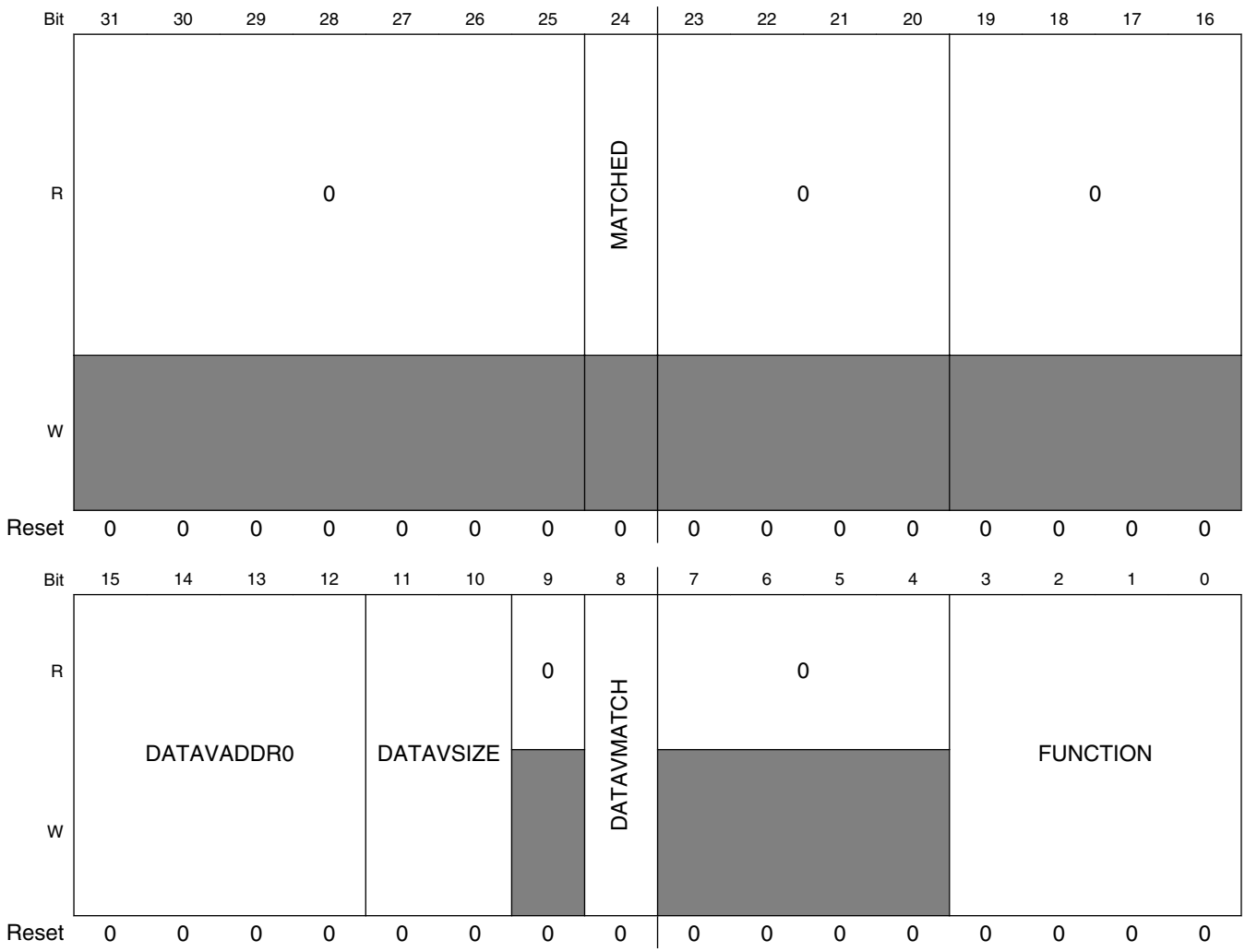
MTBDWT_MASK*n* field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MASK	MASK The value of the ignore mask, 0-31 bits, is applied to address range matching. MASK = 0 is used to include all bits of the address in the comparison, except if MASK = 0 and the comparator is configured to watch instruction fetch addresses, address bit [0] is ignored by the hardware since all fetches must be at least halfword aligned. For MASK != 0 and regardless of watch type, address bits [x-1:0] are ignored in the address comparison. Using a mask means the comparator matches on a range of addresses, defined by the unmasked most significant bits of the address, bits [31:x]. The maximum MASK value is 24, producing a 16 Mbyte mask. An attempted write of a MASK value > 24 is limited by the MTBDWT hardware to 24. If MTBDWT_COMP0 is used as a data value comparator, then MTBDWT_MASK0 should be programmed to zero.

21.3.2.4 MTB_DWT Comparator Function Register 0 (MTBDWT_FCT0)

The MTBDWT_FCTn registers control the operation of comparator n.

Address: F000_1000h base + 28h offset = F000_1028h



MTBDWT_FCT0 field descriptions

Field	Description
31–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 MATCHED	Comparator match If this read-only flag is asserted, it indicates the operation defined by the FUNCTION field occurred since the last read of the register. Reading the register clears this bit. 0 No match. 1 Match occurred.

Table continues on the next page...

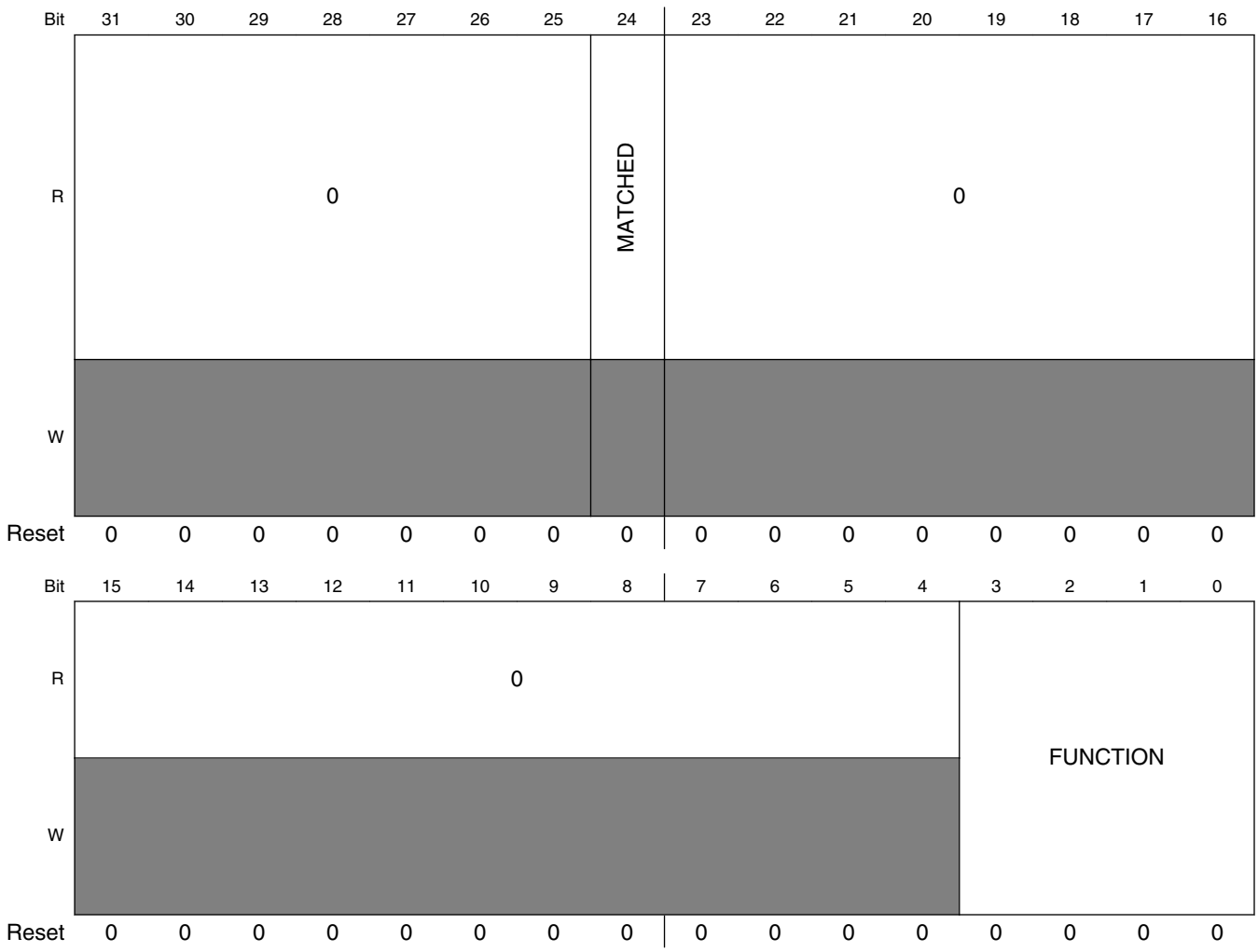
MTBDWT_FCT0 field descriptions (continued)

Field	Description
23–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–12 DATAVADDR0	Data Value Address 0 Since the MTB_DWT implements two comparators, the DATAVADDR0 field is restricted to values {0,1}. When the DATAVMATCH bit is asserted, this field defines the comparator number to use for linked address comparison. If MTBDWT_COMP0 is used as a data watchpoint and MTBDWT_COMP1 as an address watchpoint, DATAVADDR0 must be set.
11–10 DATAVSIZE	Data Value Size For data value matching, this field defines the size of the required data comparison. 00 Byte. 01 Halfword. 10 Word. 11 Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 DATAVMATCH	Data Value Match When this field is 1, it enables data value comparison. For this implementation, MTBDWT_COMP0 supports address or data value comparisons; MTBDWT_COMP1 only supports address comparisons. 0 Perform address comparison. 1 Perform data value comparison.
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FUNCTION	Function Selects the action taken on a comparator match. If MTBDWT_COMP0 is used for a data value and MTBDWT_COMP1 for an address value, then MTBDWT_FCT1[FUNCTION] must be set to zero. For this configuration, MTBDWT_MASK1 can be set to a non-zero value, so the combined comparators match on a range of addresses. 0000 Disabled. 0100 Instruction fetch. 0101 Data operand read. 0110 Data operand write. 0111 Data operand (read + write). others Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.

21.3.2.5 MTB_DWT Comparator Function Register 1 (MTBDWT_FCT1)

The MTBDWT_FCTn registers control the operation of comparator n. Since the MTB_DWT only supports data value comparisons on comparator 0, there are several fields in the MTBDWT_FCT1 register that are RAZ/WI (bits 12, 11:10, 8).

Address: F000_1000h base + 38h offset = F000_1038h



MTBDWT_FCT1 field descriptions

Field	Description
31–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 MATCHED	Comparator match If this read-only flag is asserted, it indicates the operation defined by the FUNCTION field occurred since the last read of the register. Reading the register clears this bit.

Table continues on the next page...

MTBDWT_FCT1 field descriptions (continued)

Field	Description
	0 No match. 1 Match occurred.
23–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FUNCTION	Function Selects the action taken on a comparator match. If MTBDWT_COMP0 is used for a data value and MTBDWT_COMP1 for an address value, then MTBDWT_FCT1[FUNCTION] must be set to zero. For this configuration, MTBDWT_MASK1 can be set to a non-zero value, so the combined comparators match on a range of addresses. 0000 Disabled. 0100 Instruction fetch. 0101 Data operand read. 0110 Data operand write. 0111 Data operand (read + write). others Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.

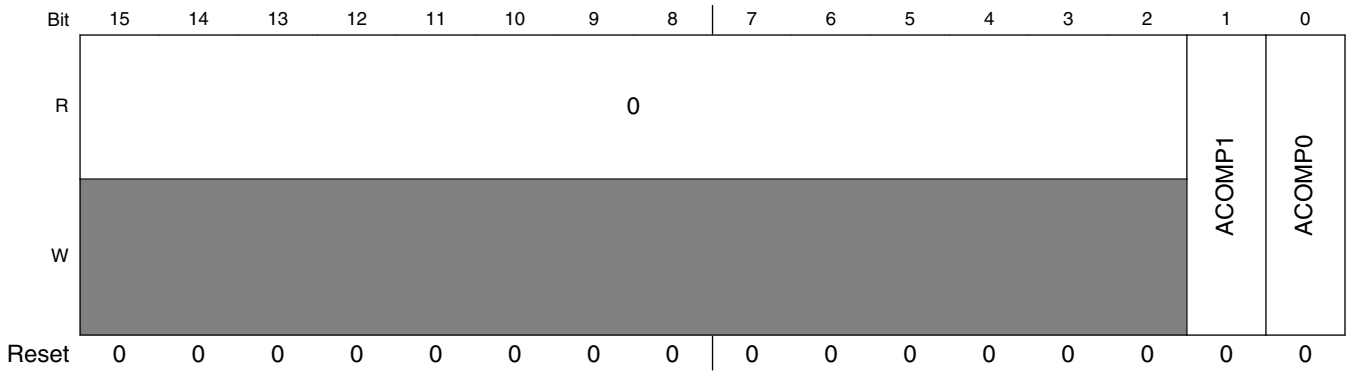
21.3.2.6 MTB_DWT Trace Buffer Control Register (MTBDWT_TBCTRL)

The MTBDWT_TBCTRL register defines how the watchpoint comparisons control the actual trace buffer operation.

Recall the MTB supports starting and stopping the program trace based on the watchpoint comparisons signaled via TSTART and TSTOP. The watchpoint comparison signals are enabled in the MTB's control logic by setting the appropriate enable bits, MTB_MASTER[TSTARTEN, TSTOPEN]. In the event of simultaneous assertion of both TSTART and TSTOP, TSTART takes priority.

Address: F000_1000h base + 200h offset = F000_1200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NUMCOMP				0											
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0



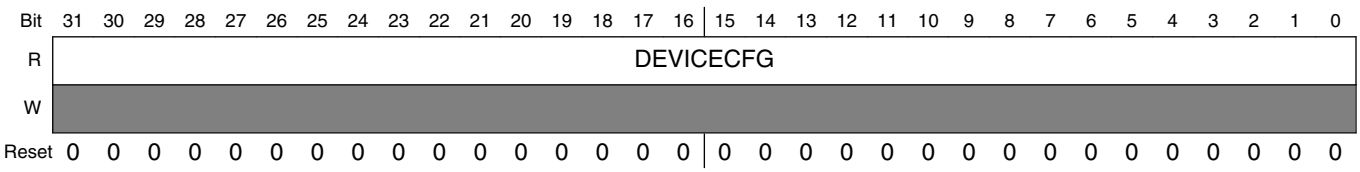
MTBDWT_TBCTRL field descriptions

Field	Description
31–28 NUMCOMP	Number of Comparators This read-only field specifies the number of comparators in the MTB_DWT. This implementation includes two registers.
27–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 ACOMP1	Action based on Comparator 1 match When the MTBDWT_FCT1[MATCHED] is set, it indicates MTBDWT_COMP1 address compare has triggered and the trace buffer's recording state is changed. 0 Trigger TSTOP based on the assertion of MTBDWT_FCT1[MATCHED]. 1 Trigger TSTART based on the assertion of MTBDWT_FCT1[MATCHED].
0 ACOMP0	Action based on Comparator 0 match When the MTBDWT_FCT0[MATCHED] is set, it indicates MTBDWT_COMP0 address compare has triggered and the trace buffer's recording state is changed. The assertion of MTBDWT_FCT0[MATCHED] is caused by the following conditions: • Address match in MTBDWT_COMP0 when MTBDWT_FCT0[DATAVMATCH] = 0 • Data match in MTBDWT_COMP0 when MTBDWT_FCT0[DATAVMATCH, DATAVADDR0] = {1,0} • Data match in MTBDWT_COMP0 and address match in MTBDWT_COMP1 when MTBDWT_FCT0[DATAVMATCH, DATAVADDR0] = {1,1} 0 Trigger TSTOP based on the assertion of MTBDWT_FCT0[MATCHED]. 1 Trigger TSTART based on the assertion of MTBDWT_FCT0[MATCHED].

21.3.2.7 Device Configuration Register (MTBDWT_DEVICECFG)

This register indicates the device configuration. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_1000h base + FC8h offset = F000_1FC8h



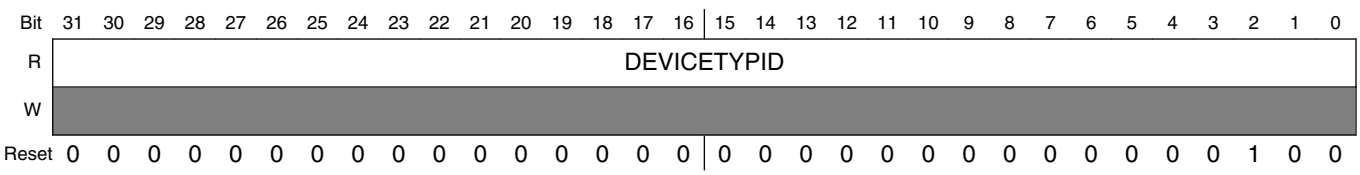
MTBDWT_DEVICECFG field descriptions

Field	Description
DEVICECFG	DEVICECFG Hardwired to 0x0000_0000.

21.3.2.8 Device Type Identifier Register (MTBDWT_DEVICETYPID)

This register indicates the device type ID. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_1000h base + FCCh offset = F000_1FCCh



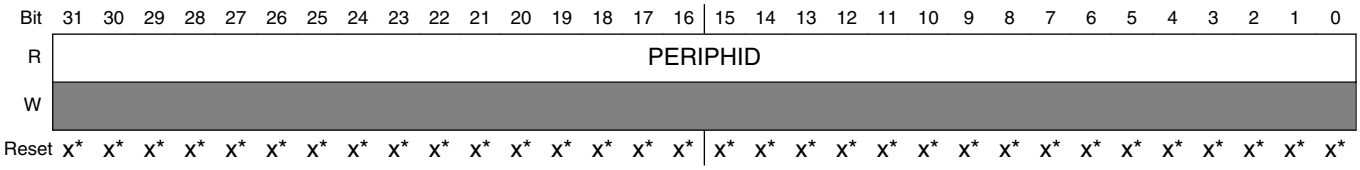
MTBDWT_DEVICETYPID field descriptions

Field	Description
DEVICETYPID	DEVICETYPID Hardwired to 0x0000_0004.

21.3.2.9 Peripheral ID Register (MTBDWT_PERIPHID*n*)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_1000h base + FD0h offset + (4d × i), where i=0d to 7d



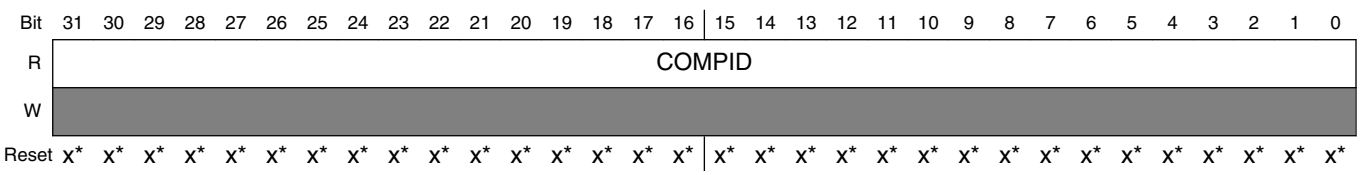
MTBDWT_PERIPHID*n* field descriptions

Field	Description
PERIPHID	PERIPHID Peripheral ID1 is hardwired to 0x0000_00E0; ID2 to 0x0000_0008; and all the others to 0x0000_0000.

21.3.2.10 Component ID Register (MTBDWT_COMPID*n*)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_1000h base + FF0h offset + (4d × i), where i=0d to 3d



MTBDWT_COMPID*n* field descriptions

Field	Description
COMPID	Component ID Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0090; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.

21.3.3 System ROM Memory Map

The System ROM Table registers are also mapped into a sparsely-populated 4 KB address space.

For core configurations like that supported by Cortex-M0+, ARM recommends that a debugger identifies and connects to the debug components using the CoreSight debug infrastructure.

ARM recommends that a debugger follows the flow as shown in the following figure to discover the components in the CoreSight debug infrastructure. In this case, a debugger reads the peripheral and component ID registers for each CoreSight component in the CoreSight system.

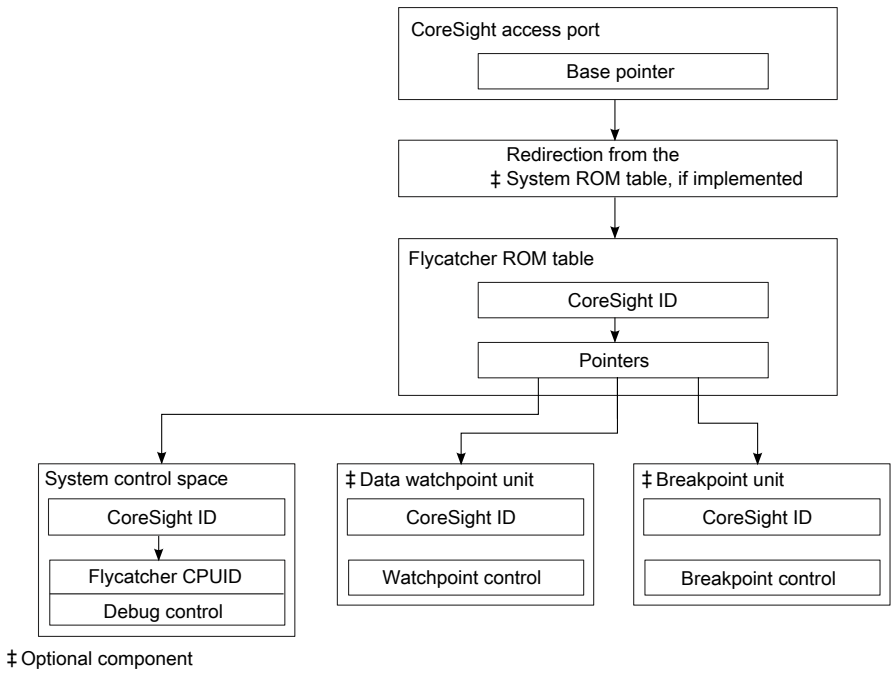


Figure 21-2. CoreSight discovery process

ROM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_2000	Entry (ROM_ENTRY0)	32	R	See section	21.3.3.1/ 424
F000_2004	Entry (ROM_ENTRY1)	32	R	See section	21.3.3.1/ 424
F000_2008	Entry (ROM_ENTRY2)	32	R	See section	21.3.3.1/ 424
F000_200C	End of Table Marker Register (ROM_TABLEMARK)	32	R	0000_0000h	21.3.3.2/ 425
F000_2FCC	System Access Register (ROM_SYSACCESS)	32	R	0000_0001h	21.3.3.3/ 425
F000_2FD0	Peripheral ID Register (ROM_PERIPHID4)	32	R	See section	21.3.3.4/ 426

Table continues on the next page...

ROM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_2FD4	Peripheral ID Register (ROM_PERIPHID5)	32	R	See section	21.3.3.4/ 426
F000_2FD8	Peripheral ID Register (ROM_PERIPHID6)	32	R	See section	21.3.3.4/ 426
F000_2FDC	Peripheral ID Register (ROM_PERIPHID7)	32	R	See section	21.3.3.4/ 426
F000_2FE0	Peripheral ID Register (ROM_PERIPHID0)	32	R	See section	21.3.3.4/ 426
F000_2FE4	Peripheral ID Register (ROM_PERIPHID1)	32	R	See section	21.3.3.4/ 426
F000_2FE8	Peripheral ID Register (ROM_PERIPHID2)	32	R	See section	21.3.3.4/ 426
F000_2FEC	Peripheral ID Register (ROM_PERIPHID3)	32	R	See section	21.3.3.4/ 426
F000_2FF0	Component ID Register (ROM_COMPID0)	32	R	See section	21.3.3.5/ 426
F000_2FF4	Component ID Register (ROM_COMPID1)	32	R	See section	21.3.3.5/ 426
F000_2FF8	Component ID Register (ROM_COMPID2)	32	R	See section	21.3.3.5/ 426
F000_2FFC	Component ID Register (ROM_COMPID3)	32	R	See section	21.3.3.5/ 426

21.3.3.1 Entry (ROM_ENTRY*n*)

The System ROM Table begins with "n" relative 32-bit addresses, one for each debug component present in the device and terminating with an all-zero value signaling the end of the table at the "n+1"-th value.

It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + 0h offset + (4d × i), where i=0d to 2d

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	ENTRY																															
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*

ROM_ENTRY*n* field descriptions

Field	Description
ENTRY	ENTRY

ROM_ENTRY*n* field descriptions (continued)

Field	Description
	Entry 0 (MTB) is hardwired to 0xFFFF_E003; Entry 1 (MTBDWT) to 0xFFFF_F003; Entry 2 (CM0+ ROM Table) to 0xF00F_D003.

21.3.3.2 End of Table Marker Register (ROM_TABLEMARK)

This register indicates end of table marker. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + Ch offset = F000_200Ch

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	MARK																															
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	0	0	0

ROM_TABLEMARK field descriptions

Field	Description
MARK	MARK Hardwired to 0x0000_0000

21.3.3.3 System Access Register (ROM_SYSACCESS)

This register indicates system access. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + FCCh offset = F000_2FCCh

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	SYSACCESS																															
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	0	0	1

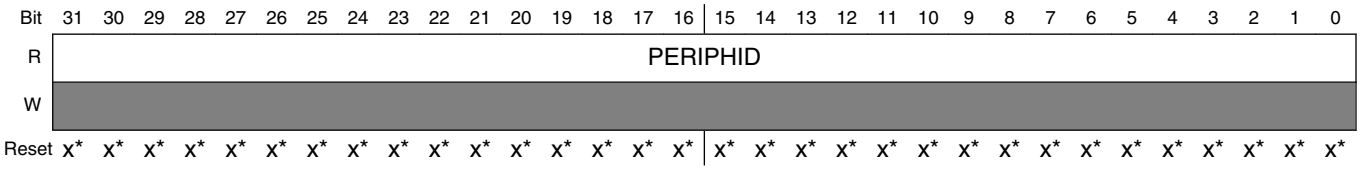
ROM_SYSACCESS field descriptions

Field	Description
SYSACCESS	SYSACCESS Hardwired to 0x0000_0001

21.3.3.4 Peripheral ID Register (ROM_PERIPHID*n*)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + FD0h offset + (4d × i), where i=0d to 7d



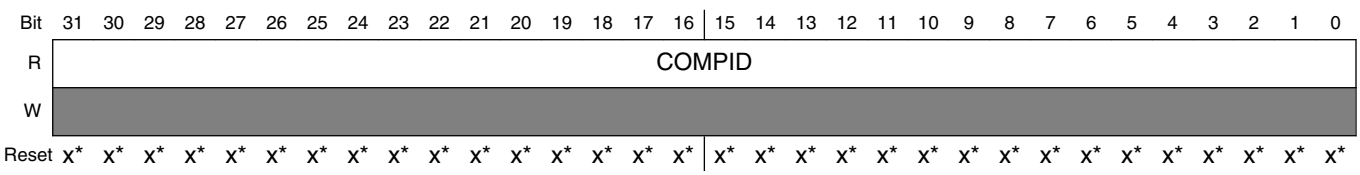
ROM_PERIPHID*n* field descriptions

Field	Description
PERIPHID	PERIPHID Peripheral ID1 is hardwired to 0x0000_00E0; ID2 to 0x0000_0008; and all the others to 0x0000_0000.

21.3.3.5 Component ID Register (ROM_COMPID*n*)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + FF0h offset + (4d × i), where i=0d to 3d



ROM_COMPID*n* field descriptions

Field	Description
COMPID	Component ID Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0010; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.

Chapter 22
Crossbar Switch Lite (AXBS-Lite)

22.1 Introduction

NOTE

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

The information found here provides information on the layout, configuration, and programming of the crossbar switch.

The crossbar switch connects bus masters and bus slaves using a crossbar switch structure. This structure allows up to four bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave.

22.1.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
  - Allows concurrent accesses from different masters to different slaves
- Up to single-clock 32-bit transfer
- Programmable configuration for fixed-priority or round-robin slave port arbitration (see the chip-specific information).

22.2 Memory Map / Register Definition

This crossbar switch is designed for minimal gate count. It, therefore, has no memory-mapped configuration registers.

Please see the chip-specific information for information on whether the arbitration method in the crossbar switch is programmable, and by which module.

22.3 Functional Description

22.3.1 General operation

When a master accesses the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply waits.

After the master has control of the slave port it is targeting, the master remains in control of the slave port until it relinquishes the slave port by running an IDLE cycle or by targeting a different slave port for its next access.

The master can also lose control of the slave port if another higher-priority master makes a request to the slave port.

The crossbar terminates all master IDLE transfers, as opposed to allowing the termination to come from one of the slave buses. Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it remains parked with the last master to use the slave port. This is done to save the initial clock of arbitration delay that otherwise would be seen if the same master had to arbitrate to gain control of the slave port.

22.3.2 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round-robin

The selection of the global slave port arbitration is controlled in the MCM module. For fixed priority, set MCM_PLACR[ARB] to 0. For round robin, set MCM_PLACR[ARB] to 1. This arbitration setting applies to all slave ports.

22.3.2.1 Arbitration during undefined length bursts

All lengths of burst accesses lock out arbitration until the last beat of the burst.

22.3.2.2 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level with the highest numbered master having the highest priority (for example, in a system with 5 masters, master 1 has lower priority than master 3). If two masters request access to the same slave port, the master with the highest priority gains control over the slave port.

NOTE

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

Table 22-1. How the Crossbar Switch grants control of a slave port to a master

When	Then the Crossbar Switch grants control to the requesting master
Both of the following are true:	At the next clock edge

Table continues on the next page...

Table 22-1. How the Crossbar Switch grants control of a slave port to a master (continued)

When	Then the Crossbar Switch grants control to the requesting master
• The current master is not running a transfer. • The new requesting master's priority level is higher than that of the current master.	
The requesting master's priority level is lower than the current master.	At the conclusion of one of the following cycles: • An IDLE cycle • A non-IDLE cycle to a location other than the current slave port

22.3.2.3 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

The round-robin arbitration mode generally provides a more fair allocation of the available slave-port bandwidth (compared to fixed priority) as the fixed master priority does not affect the master selection.

22.4 Initialization/application information

No initialization is required for the crossbar switch. See the chip-specific crossbar switch information for the reset state of the arbitration scheme.

Chapter 23
Peripheral Bridge (AIPS-Lite)

23.1 Introduction

NOTE

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

The peripheral bridge converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

The peripheral bridge occupies 64 MB of the address space, which is divided into peripheral slots of 4 KB. (It might be possible that all the peripheral slots are not used. See the memory map chapter for details on slot assignments.) The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

23.1.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width

23.1.2 General operation

The slave devices connected to the peripheral bridge are modules which contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge.

The register maps of the peripherals are located on 4-KB boundaries. Each peripheral is allocated one or more 4-KB block(s) of the memory map.

23.2 Memory map/register definition

The AIPS module(s) on this device do(es) not contain any user-programmable registers.

23.3 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus.

The peripheral bridge manages all transactions destined for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

23.3.1 Access support

All combinations of access size and peripheral data port width are supported. An access that is larger than the target peripheral's data width will be decomposed to multiple, smaller accesses. Bus decomposition is terminated by a transfer error caused by an access to an empty register area.

Chapter 24
Direct Memory Access Multiplexer (DMAMUX)

24.1 Introduction

NOTE

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

24.1.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 8 DMA channels. This process is illustrated in the following figure.

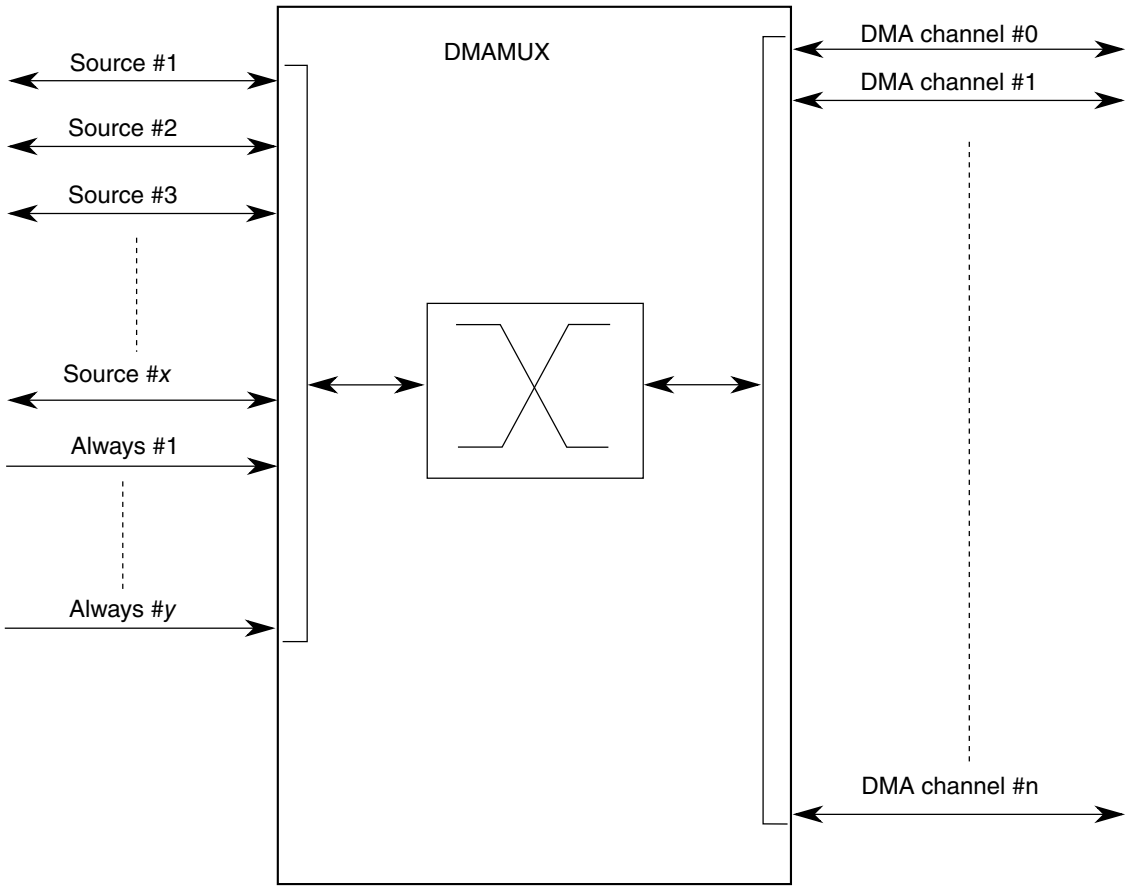


Figure 24-1. DMAMUX block diagram

24.1.2 Features

The DMAMUX module provides these features:

- Up to 63 peripheral slots and up to six always-on slots can be routed to 8 channels.
- 8 independently selectable DMA channel routers.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

24.1.3 Modes of operation

The following operating modes are available:

- Disabled mode

In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

24.2 External signal description

The DMAMUX has no external pins.

24.3 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

DMAMUX memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_1000	Channel Configuration register (DMAMUX_CHCFG0)	8	R/W	00h	24.3.1/435
4002_1001	Channel Configuration register (DMAMUX_CHCFG1)	8	R/W	00h	24.3.1/435
4002_1002	Channel Configuration register (DMAMUX_CHCFG2)	8	R/W	00h	24.3.1/435
4002_1003	Channel Configuration register (DMAMUX_CHCFG3)	8	R/W	00h	24.3.1/435
4002_1004	Channel Configuration register (DMAMUX_CHCFG4)	8	R/W	00h	24.3.1/435
4002_1005	Channel Configuration register (DMAMUX_CHCFG5)	8	R/W	00h	24.3.1/435
4002_1006	Channel Configuration register (DMAMUX_CHCFG6)	8	R/W	00h	24.3.1/435
4002_1007	Channel Configuration register (DMAMUX_CHCFG7)	8	R/W	00h	24.3.1/435

24.3.1 Channel Configuration register (DMAMUX_CHCFG*n*)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

NOTE

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the source settings, a DMA channel must be disabled via CHCFGn[ENBL].

Address: 4002_1000h base + 0h offset + (1d × i), where i=0d to 7d

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

DMAMUX_CHCFG*n* field descriptions

Field	Description
7 ENBL	DMA Channel Enable Enables the DMA channel. 0 DMA channel is disabled. This mode is primarily used during configuration of the DMAMux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel. 1 DMA channel is enabled
6 Reserved	This field is reserved. This read/write field does not affect the functionality of the device and should not be used.
SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about the peripherals and their slot numbers.

24.4 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

The DMAMUX channels implement only the normal routing functionality.

24.4.1 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are six additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins.
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

  By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.

  In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers *after every minor loop*. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use an always-enabled DMA source.

  In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source.

24.5 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

24.5.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

24.5.2 Enabling and configuring sources

To enable a source, the following steps can be used:

1. Determine with which DMA channel the source will be associated.
2. Clear the CHCFG[ENBL] field of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set.

To configure source #5 transmit for use with DMA channel 1, as an example, the following steps can be used:

1. Write 0x00 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
```

```
In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;
*CHCFG1 = 0x85;
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] bit of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] field is set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCFG8.
3. Write 0x87 to CHCFG8.

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);

In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00;
*CHCFG8 = 0x87;
```

Chapter 25
Enhanced Direct Memory Access (eDMA)

25.1 Introduction

NOTE

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

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 8 channels

25.1.1 eDMA system block diagram

Figure 25-1 illustrates the components of the eDMA system, including the eDMA module ("engine").

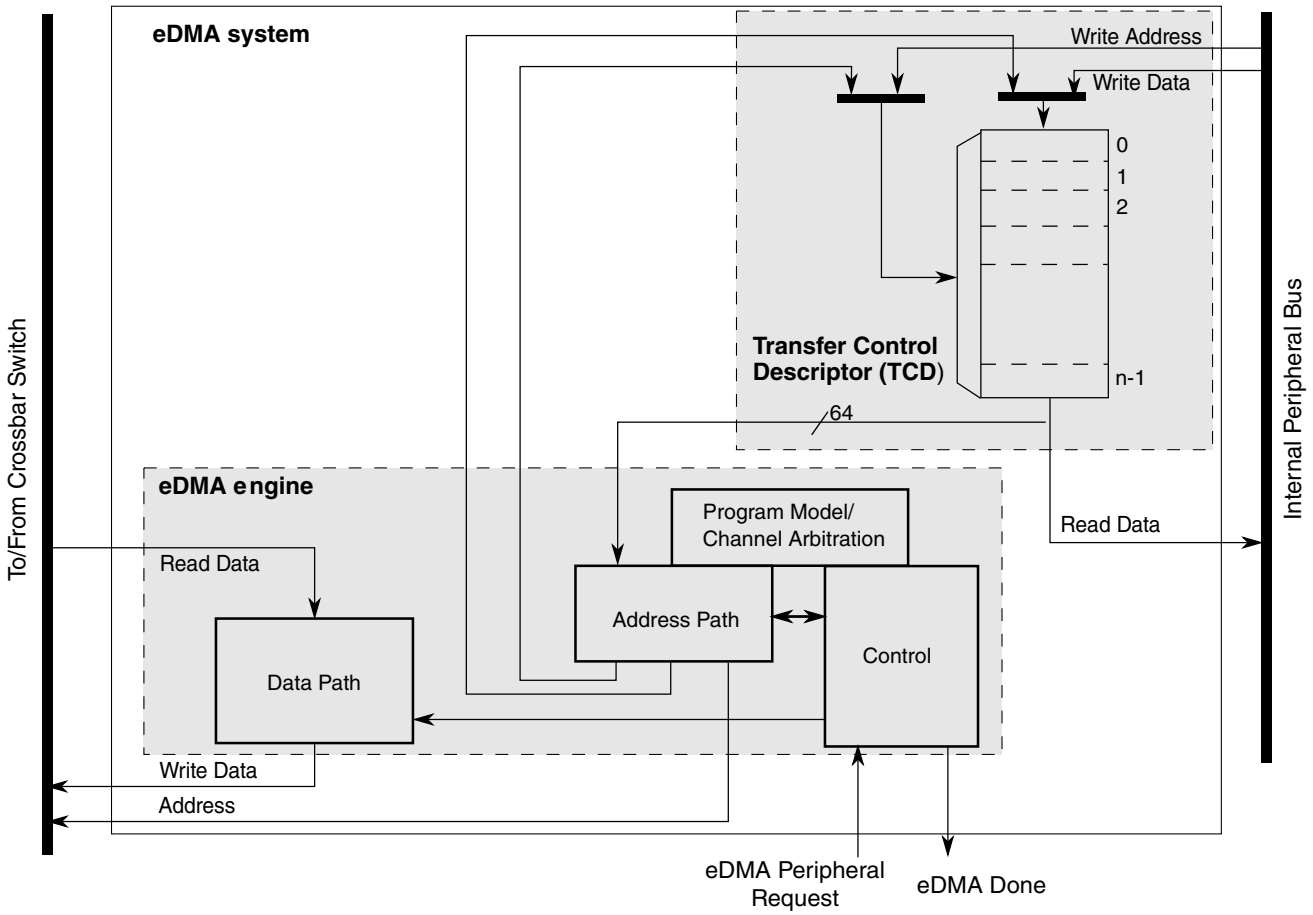


Figure 25-1. eDMA system block diagram

25.1.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

Table 25-1. eDMA engine submodules

Submodule	Function
Address path	This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution. When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes

Table continues on the next page...

Table 25-1. eDMA engine submodules (continued)

Submodule	Function
	the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	This block implements the bus master read/write datapath. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase).
Program model/channel arbitration	This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
Control	This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

The transfer-control descriptor local memory is further partitioned into:

Table 25-2. Transfer control descriptor memory

Submodule	Description
Memory controller	This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
Memory array	TCD storage for each channel's transfer profile.

25.1.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes

- 8-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via programmable interrupt requests
  - One interrupt per channel, which can be asserted at completion of major iteration count
  - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing
- Support for complex data structures

In the discussion of this module, *n* is used to reference the channel number.

25.2 Modes of operation

The eDMA operates in the following modes:

Table 25-3. Modes of operation

Mode	Description
Normal	In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA.

Table continues on the next page...

Table 25-3. Modes of operation (continued)

Mode	Description
	A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.
Debug	DMA operation is configurable in Debug mode via the control register: • If CR[EDBG] is cleared, the DMA continues to operate. • If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.
Wait	Before entering Wait mode, the DMA attempts to complete its current transfer. After the transfer completes, the device enters Wait mode.

25.3 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions
- The second region corresponds to the local transfer control descriptor (TCD) memory

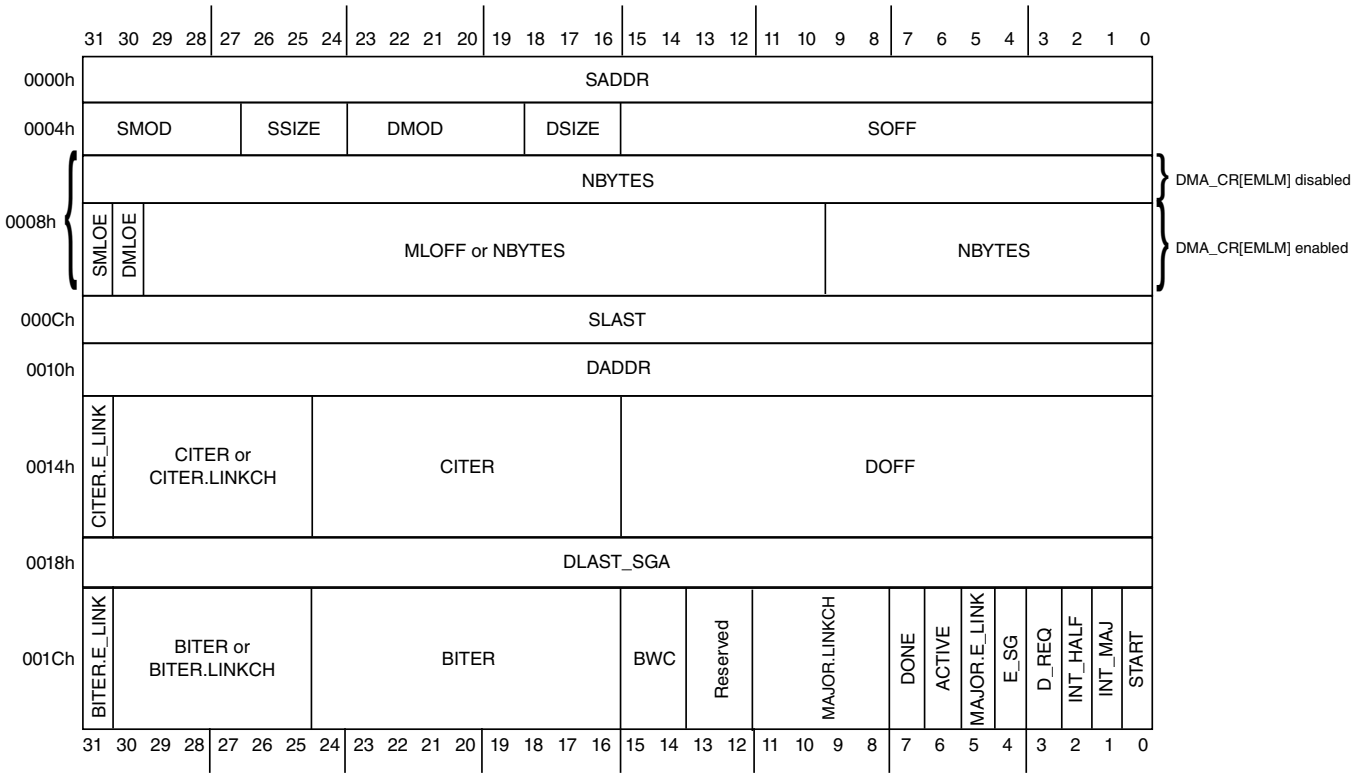
25.3.1 TCD memory

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel 7. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

25.3.2 TCD initialization

Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

25.3.3 TCD structure



25.3.4 Reserved memory and bit fields

- Reading reserved bits in a register returns the value of zero.
- Writes to reserved bits in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

DMA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_8000	Control Register (DMA_CR)	32	R/W	See section	25.3.5/452
4000_8004	Error Status Register (DMA_ES)	32	R	0000_0000h	25.3.6/455
4000_800C	Enable Request Register (DMA_ERQ)	32	R/W	0000_0000h	25.3.7/457
4000_8014	Enable Error Interrupt Register (DMA_EEI)	32	R/W	0000_0000h	25.3.8/459
4000_8018	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	25.3.9/460
4000_8019	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	25.3.10/461

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_801A	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	25.3.11/462
4000_801B	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	25.3.12/463
4000_801C	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	25.3.13/464
4000_801D	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	25.3.14/465
4000_801E	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	25.3.15/466
4000_801F	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	25.3.16/467
4000_8024	Interrupt Request Register (DMA_INT)	32	R/W	0000_0000h	25.3.17/468
4000_802C	Error Register (DMA_ERR)	32	R/W	0000_0000h	25.3.18/469
4000_8034	Hardware Request Status Register (DMA_HRS)	32	R	0000_0000h	25.3.19/471
4000_8044	Enable Asynchronous Request in Stop Register (DMA_EARS)	32	R/W	0000_0000h	25.3.20/473
4000_8100	Channel n Priority Register (DMA_DCHPRI3)	8	R/W	See section	25.3.21/474
4000_8101	Channel n Priority Register (DMA_DCHPRI2)	8	R/W	See section	25.3.21/474
4000_8102	Channel n Priority Register (DMA_DCHPRI1)	8	R/W	See section	25.3.21/474
4000_8103	Channel n Priority Register (DMA_DCHPRI0)	8	R/W	See section	25.3.21/474
4000_8104	Channel n Priority Register (DMA_DCHPRI7)	8	R/W	See section	25.3.21/474
4000_8105	Channel n Priority Register (DMA_DCHPRI6)	8	R/W	See section	25.3.21/474
4000_8106	Channel n Priority Register (DMA_DCHPRI5)	8	R/W	See section	25.3.21/474
4000_8107	Channel n Priority Register (DMA_DCHPRI4)	8	R/W	See section	25.3.21/474
4000_9000	TCD Source Address (DMA_TCD0_SADDR)	32	R/W	Undefined	25.3.22/475
4000_9004	TCD Signed Source Address Offset (DMA_TCD0_SOFF)	16	R/W	Undefined	25.3.23/475
4000_9006	TCD Transfer Attributes (DMA_TCD0_ATTR)	16	R/W	Undefined	25.3.24/476
4000_9008	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD0_NBYTES_MLNO)	32	R/W	Undefined	25.3.25/477
4000_9008	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD0_NBYTES_MLOFFNO)	32	R/W	Undefined	25.3.26/477
4000_9008	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD0_NBYTES_MLOFFYES)	32	R/W	Undefined	25.3.27/479
4000_900C	TCD Last Source Address Adjustment (DMA_TCD0_SLAST)	32	R/W	Undefined	25.3.28/480

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_9010	TCD Destination Address (DMA_TCD0_DADDR)	32	R/W	Undefined	25.3.29/480
4000_9014	TCD Signed Destination Address Offset (DMA_TCD0_DOFF)	16	R/W	Undefined	25.3.30/481
4000_9016	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_CITER_ELINKYES)	16	R/W	Undefined	25.3.31/481
4000_9016	DMA_TCD0_CITER_ELINKNO	16	R/W	Undefined	25.3.32/483
4000_9018	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD0_DLASTSGA)	32	R/W	Undefined	25.3.33/484
4000_901C	TCD Control and Status (DMA_TCD0_CSR)	16	R/W	Undefined	25.3.34/484
4000_901E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_BITER_ELINKYES)	16	R/W	Undefined	25.3.35/487
4000_901E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_BITER_ELINKNO)	16	R/W	Undefined	25.3.36/488
4000_9020	TCD Source Address (DMA_TCD1_SADDR)	32	R/W	Undefined	25.3.22/475
4000_9024	TCD Signed Source Address Offset (DMA_TCD1_SOFF)	16	R/W	Undefined	25.3.23/475
4000_9026	TCD Transfer Attributes (DMA_TCD1_ATTR)	16	R/W	Undefined	25.3.24/476
4000_9028	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD1_NBYTES_MLNO)	32	R/W	Undefined	25.3.25/477
4000_9028	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD1_NBYTES_MLOFFNO)	32	R/W	Undefined	25.3.26/477
4000_9028	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD1_NBYTES_MLOFFYES)	32	R/W	Undefined	25.3.27/479
4000_902C	TCD Last Source Address Adjustment (DMA_TCD1_SLAST)	32	R/W	Undefined	25.3.28/480
4000_9030	TCD Destination Address (DMA_TCD1_DADDR)	32	R/W	Undefined	25.3.29/480
4000_9034	TCD Signed Destination Address Offset (DMA_TCD1_DOFF)	16	R/W	Undefined	25.3.30/481
4000_9036	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_CITER_ELINKYES)	16	R/W	Undefined	25.3.31/481
4000_9036	DMA_TCD1_CITER_ELINKNO	16	R/W	Undefined	25.3.32/483
4000_9038	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD1_DLASTSGA)	32	R/W	Undefined	25.3.33/484
4000_903C	TCD Control and Status (DMA_TCD1_CSR)	16	R/W	Undefined	25.3.34/484
4000_903E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_BITER_ELINKYES)	16	R/W	Undefined	25.3.35/487
4000_903E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_BITER_ELINKNO)	16	R/W	Undefined	25.3.36/488
4000_9040	TCD Source Address (DMA_TCD2_SADDR)	32	R/W	Undefined	25.3.22/475
4000_9044	TCD Signed Source Address Offset (DMA_TCD2_SOFF)	16	R/W	Undefined	25.3.23/475
4000_9046	TCD Transfer Attributes (DMA_TCD2_ATTR)	16	R/W	Undefined	25.3.24/476

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_9048	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD2_NBYTES_MLNO)	32	R/W	Undefined	25.3.25/477
4000_9048	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD2_NBYTES_MLOFFNO)	32	R/W	Undefined	25.3.26/477
4000_9048	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD2_NBYTES_MLOFFYES)	32	R/W	Undefined	25.3.27/479
4000_904C	TCD Last Source Address Adjustment (DMA_TCD2_SLAST)	32	R/W	Undefined	25.3.28/480
4000_9050	TCD Destination Address (DMA_TCD2_DADDR)	32	R/W	Undefined	25.3.29/480
4000_9054	TCD Signed Destination Address Offset (DMA_TCD2_DOFF)	16	R/W	Undefined	25.3.30/481
4000_9056	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_CITER_ELINKYES)	16	R/W	Undefined	25.3.31/481
4000_9056	DMA_TCD2_CITER_ELINKNO	16	R/W	Undefined	25.3.32/483
4000_9058	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD2_DLASTSGA)	32	R/W	Undefined	25.3.33/484
4000_905C	TCD Control and Status (DMA_TCD2_CSR)	16	R/W	Undefined	25.3.34/484
4000_905E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_BITER_ELINKYES)	16	R/W	Undefined	25.3.35/487
4000_905E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_BITER_ELINKNO)	16	R/W	Undefined	25.3.36/488
4000_9060	TCD Source Address (DMA_TCD3_SADDR)	32	R/W	Undefined	25.3.22/475
4000_9064	TCD Signed Source Address Offset (DMA_TCD3_SOFF)	16	R/W	Undefined	25.3.23/475
4000_9066	TCD Transfer Attributes (DMA_TCD3_ATTR)	16	R/W	Undefined	25.3.24/476
4000_9068	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD3_NBYTES_MLNO)	32	R/W	Undefined	25.3.25/477
4000_9068	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD3_NBYTES_MLOFFNO)	32	R/W	Undefined	25.3.26/477
4000_9068	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD3_NBYTES_MLOFFYES)	32	R/W	Undefined	25.3.27/479
4000_906C	TCD Last Source Address Adjustment (DMA_TCD3_SLAST)	32	R/W	Undefined	25.3.28/480
4000_9070	TCD Destination Address (DMA_TCD3_DADDR)	32	R/W	Undefined	25.3.29/480
4000_9074	TCD Signed Destination Address Offset (DMA_TCD3_DOFF)	16	R/W	Undefined	25.3.30/481
4000_9076	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_CITER_ELINKYES)	16	R/W	Undefined	25.3.31/481
4000_9076	DMA_TCD3_CITER_ELINKNO	16	R/W	Undefined	25.3.32/483
4000_9078	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD3_DLASTSGA)	32	R/W	Undefined	25.3.33/484

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_907C	TCD Control and Status (DMA_TCD3_CSR)	16	R/W	Undefined	25.3.34/484
4000_907E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_BITER_ELINKYES)	16	R/W	Undefined	25.3.35/487
4000_907E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_BITER_ELINKNO)	16	R/W	Undefined	25.3.36/488
4000_9080	TCD Source Address (DMA_TCD4_SADDR)	32	R/W	Undefined	25.3.22/475
4000_9084	TCD Signed Source Address Offset (DMA_TCD4_SOFF)	16	R/W	Undefined	25.3.23/475
4000_9086	TCD Transfer Attributes (DMA_TCD4_ATTR)	16	R/W	Undefined	25.3.24/476
4000_9088	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD4_NBYTES_MLNO)	32	R/W	Undefined	25.3.25/477
4000_9088	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD4_NBYTES_MLOFFNO)	32	R/W	Undefined	25.3.26/477
4000_9088	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD4_NBYTES_MLOFFYES)	32	R/W	Undefined	25.3.27/479
4000_908C	TCD Last Source Address Adjustment (DMA_TCD4_SLAST)	32	R/W	Undefined	25.3.28/480
4000_9090	TCD Destination Address (DMA_TCD4_DADDR)	32	R/W	Undefined	25.3.29/480
4000_9094	TCD Signed Destination Address Offset (DMA_TCD4_DOFF)	16	R/W	Undefined	25.3.30/481
4000_9096	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_CITER_ELINKYES)	16	R/W	Undefined	25.3.31/481
4000_9096	DMA_TCD4_CITER_ELINKNO	16	R/W	Undefined	25.3.32/483
4000_9098	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD4_DLASTSGA)	32	R/W	Undefined	25.3.33/484
4000_909C	TCD Control and Status (DMA_TCD4_CSR)	16	R/W	Undefined	25.3.34/484
4000_909E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_BITER_ELINKYES)	16	R/W	Undefined	25.3.35/487
4000_909E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD4_BITER_ELINKNO)	16	R/W	Undefined	25.3.36/488
4000_90A0	TCD Source Address (DMA_TCD5_SADDR)	32	R/W	Undefined	25.3.22/475
4000_90A4	TCD Signed Source Address Offset (DMA_TCD5_SOFF)	16	R/W	Undefined	25.3.23/475
4000_90A6	TCD Transfer Attributes (DMA_TCD5_ATTR)	16	R/W	Undefined	25.3.24/476
4000_90A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD5_NBYTES_MLNO)	32	R/W	Undefined	25.3.25/477
4000_90A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD5_NBYTES_MLOFFNO)	32	R/W	Undefined	25.3.26/477
4000_90A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD5_NBYTES_MLOFFYES)	32	R/W	Undefined	25.3.27/479

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_90AC	TCD Last Source Address Adjustment (DMA_TCD5_SLAST)	32	R/W	Undefined	25.3.28/480
4000_90B0	TCD Destination Address (DMA_TCD5_DADDR)	32	R/W	Undefined	25.3.29/480
4000_90B4	TCD Signed Destination Address Offset (DMA_TCD5_DOFF)	16	R/W	Undefined	25.3.30/481
4000_90B6	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_CITER_ELINKYES)	16	R/W	Undefined	25.3.31/481
4000_90B6	DMA_TCD5_CITER_ELINKNO	16	R/W	Undefined	25.3.32/483
4000_90B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD5_DLASTSGA)	32	R/W	Undefined	25.3.33/484
4000_90BC	TCD Control and Status (DMA_TCD5_CSR)	16	R/W	Undefined	25.3.34/484
4000_90BE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_BITER_ELINKYES)	16	R/W	Undefined	25.3.35/487
4000_90BE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD5_BITER_ELINKNO)	16	R/W	Undefined	25.3.36/488
4000_90C0	TCD Source Address (DMA_TCD6_SADDR)	32	R/W	Undefined	25.3.22/475
4000_90C4	TCD Signed Source Address Offset (DMA_TCD6_SOFF)	16	R/W	Undefined	25.3.23/475
4000_90C6	TCD Transfer Attributes (DMA_TCD6_ATTR)	16	R/W	Undefined	25.3.24/476
4000_90C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD6_NBYTES_MLNO)	32	R/W	Undefined	25.3.25/477
4000_90C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD6_NBYTES_MLOFFNO)	32	R/W	Undefined	25.3.26/477
4000_90C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD6_NBYTES_MLOFFYES)	32	R/W	Undefined	25.3.27/479
4000_90CC	TCD Last Source Address Adjustment (DMA_TCD6_SLAST)	32	R/W	Undefined	25.3.28/480
4000_90D0	TCD Destination Address (DMA_TCD6_DADDR)	32	R/W	Undefined	25.3.29/480
4000_90D4	TCD Signed Destination Address Offset (DMA_TCD6_DOFF)	16	R/W	Undefined	25.3.30/481
4000_90D6	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_CITER_ELINKYES)	16	R/W	Undefined	25.3.31/481
4000_90D6	DMA_TCD6_CITER_ELINKNO	16	R/W	Undefined	25.3.32/483
4000_90D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD6_DLASTSGA)	32	R/W	Undefined	25.3.33/484
4000_90DC	TCD Control and Status (DMA_TCD6_CSR)	16	R/W	Undefined	25.3.34/484
4000_90DE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_BITER_ELINKYES)	16	R/W	Undefined	25.3.35/487
4000_90DE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD6_BITER_ELINKNO)	16	R/W	Undefined	25.3.36/488
4000_90E0	TCD Source Address (DMA_TCD7_SADDR)	32	R/W	Undefined	25.3.22/475

Table continues on the next page...

DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4000_90E4	TCD Signed Source Address Offset (DMA_TCD7_SOFF)	16	R/W	Undefined	25.3.23/475
4000_90E6	TCD Transfer Attributes (DMA_TCD7_ATTR)	16	R/W	Undefined	25.3.24/476
4000_90E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD7_NBYTES_MLNO)	32	R/W	Undefined	25.3.25/477
4000_90E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD7_NBYTES_MLOFFNO)	32	R/W	Undefined	25.3.26/477
4000_90E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD7_NBYTES_MLOFFYES)	32	R/W	Undefined	25.3.27/479
4000_90EC	TCD Last Source Address Adjustment (DMA_TCD7_SLAST)	32	R/W	Undefined	25.3.28/480
4000_90F0	TCD Destination Address (DMA_TCD7_DADDR)	32	R/W	Undefined	25.3.29/480
4000_90F4	TCD Signed Destination Address Offset (DMA_TCD7_DOFF)	16	R/W	Undefined	25.3.30/481
4000_90F6	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_CITER_ELINKYES)	16	R/W	Undefined	25.3.31/481
4000_90F6	DMA_TCD7_CITER_ELINKNO	16	R/W	Undefined	25.3.32/483
4000_90F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD7_DLASTSGA)	32	R/W	Undefined	25.3.33/484
4000_90FC	TCD Control and Status (DMA_TCD7_CSR)	16	R/W	Undefined	25.3.34/484
4000_90FE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_BITER_ELINKYES)	16	R/W	Undefined	25.3.35/487
4000_90FE	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD7_BITER_ELINKNO)	16	R/W	Undefined	25.3.36/488

25.3.5 Control Register (DMA_CR)

The CR defines the basic operating configuration of the DMA.

Arbitration can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels are cycled through (from high to low channel number) without regard to priority.

NOTE

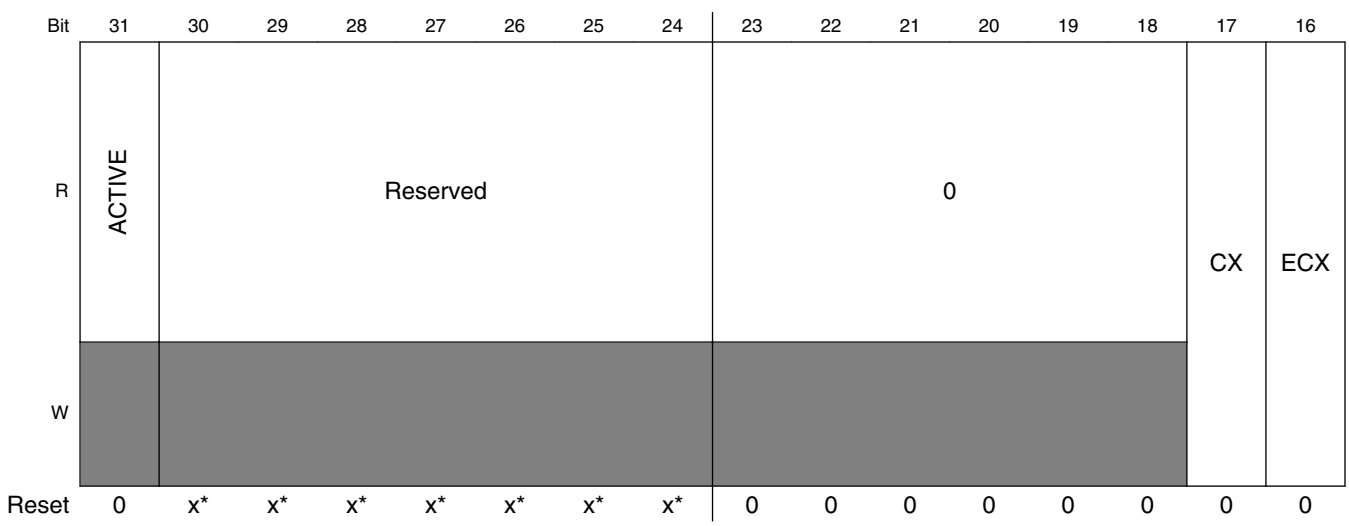
For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

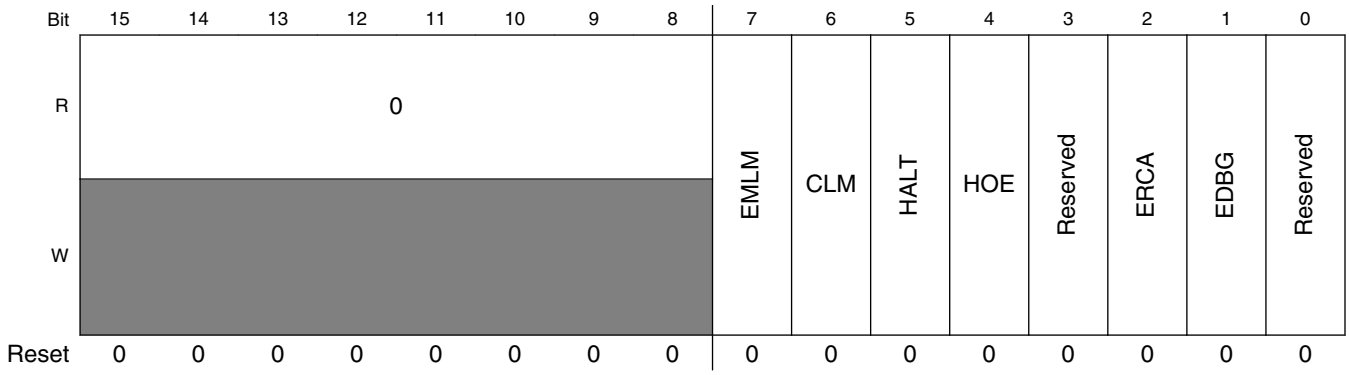
Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

Address: 4000_8000h base + 0h offset = 4000_8000h





* Notes:
- x = Undefined at reset.

DMA_CR field descriptions

Field	Description
31 ACTIVE	DMA Active Status 0 eDMA is idle. 1 eDMA is executing a channel.
30–24 Reserved	This field is reserved. Reserved
23–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 CX	Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
16 ECX	Error Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt.
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 EMLM	Enable Minor Loop Mapping 0 Disabled. TCDn.word2 is defined as a 32-bit NBYTES field. 1 Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.
6 CLM	Continuous Link Mode

Table continues on the next page...

DMA_CR field descriptions (continued)

Field	Description
	NOTE: Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, e.g., if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
5 HALT	Halt DMA Operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
4 HOE	Halt On Error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
3 Reserved	This field is reserved. Reserved
2 ERCA	Enable Round Robin Channel Arbitration 0 Fixed priority arbitration is used for channel selection . 1 Round robin arbitration is used for channel selection .
1 EDBG	Enable Debug 0 When in debug mode, the DMA continues to operate. 1 When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared.
0 Reserved	This field is reserved. Reserved

25.3.6 Error Status Register (DMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor, or
  - An illegal priority register setting in fixed-arbitration
- An error termination to a bus master read or write cycle
- A cancel transfer with error bit that will be set when a transfer is canceled via the corresponding cancel transfer control bit

See Fault reporting and handling for more details.

Memory map/register definition

Address: 4000_8000h base + 4h offset = 4000_8004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VLD	0														ECX
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	CPE	0			ERRCHN			SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ES field descriptions

Field	Description
31 VLD	Logical OR of all ERR status bits 0 No ERR bits are set. 1 At least one ERR bit is set indicating a valid error exists that has not been cleared.
30–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ECX	Transfer Canceled 0 No canceled transfers 1 The last recorded entry was a canceled transfer by the error cancel transfer input
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CPE	Channel Priority Error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities . Channel priorities are not unique.
13–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error, excluding CPE errors, or last recorded error canceled transfer.
7 SAE	Source Address Error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].
6 SOE	Source Offset Error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
5 DAE	Destination Address Error

Table continues on the next page...

DMA_ES field descriptions (continued)

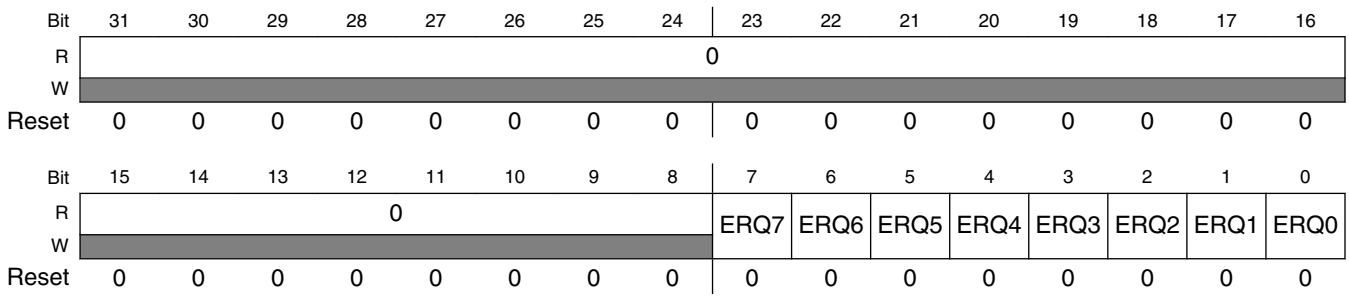
Field	Description
	0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
4 DOE	Destination Offset Error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
3 NCE	NBYTES/CITER Configuration Error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or • TCDn_CITER[CITER] is equal to zero, or • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]
2 SGE	Scatter/Gather Configuration Error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32 byte boundary.
1 SBE	Source Bus Error 0 No source bus error 1 The last recorded error was a bus error on a source read
0 DBE	Destination Bus Error 0 No destination bus error 1 The last recorded error was a bus error on a destination write

25.3.7 Enable Request Register (DMA_ERQ)

The ERQ register provides a bit map for the 8 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 4000_8000h base + Ch offset = 4000_800Ch



DMA_ERQ field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ6	Enable DMA Request 6 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ4	Enable DMA Request 4 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
0 ERQ0	Enable DMA Request 0 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

25.3.8 Enable Error Interrupt Register (DMA_EEI)

The EEI register provides a bit map for the 8 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 4000_8000h base + 14h offset = 4000_8014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_EEI field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 EEI7	Enable Error Interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI6	Enable Error Interrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI5	Enable Error Interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI4	Enable Error Interrupt 4 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI3	Enable Error Interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEI field descriptions (continued)

Field	Description
2 EEI2	Enable Error Interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI1	Enable Error Interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
0 EEI0	Enable Error Interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

25.3.9 Clear Enable Error Interrupt Register (DMA_CEEI)

The CEEI provides a simple memory-mapped mechanism to clear a given bit in the EEI to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI contents to be cleared, disabling all DMA request inputs. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 18h offset = 4000_8018h

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

DMA_CEEI field descriptions

Field	Description
7 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 CAEE	Clear All Enable Error Interrupts 0 Clear only the EEI bit specified in the CEEI field 1 Clear all bits in EEI
5–3 Reserved	This field is reserved.

Table continues on the next page...

DMA_CEEI field descriptions (continued)

Field	Description
CEEI	Clear Enable Error Interrupt Clears the corresponding bit in EEI

25.3.10 Set Enable Error Interrupt Register (DMA_SEEI)

The SEEI provides a simple memory-mapped mechanism to set a given bit in the EEI to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 19h offset = 4000_8019h

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

DMA_SEEI field descriptions

Field	Description
7 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 SAEE	Sets All Enable Error Interrupts 0 Set only the EEI bit specified in the SEEI field. 1 Sets all bits in EEI
5–3 Reserved	This field is reserved.
SEEI	Set Enable Error Interrupt Sets the corresponding bit in EEI

25.3.11 Clear Enable Request Register (DMA_CERQ)

The CERQ provides a simple memory-mapped mechanism to clear a given bit in the ERQ to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ to be cleared, disabling all DMA request inputs. If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Ah offset = 4000_801Ah

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

DMA_CERQ field descriptions

Field	Description
7 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 CAER	Clear All Enable Requests 0 Clear only the ERQ bit specified in the CERQ field 1 Clear all bits in ERQ
5–3 Reserved	This field is reserved.
CERQ	Clear Enable Request Clears the corresponding bit in ERQ.

25.3.12 Set Enable Request Register (DMA_SERQ)

The SERQ provides a simple memory-mapped mechanism to set a given bit in the ERQ to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Bh offset = 4000_801Bh

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

DMA_SERQ field descriptions

Field	Description
7 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 SAER	Set All Enable Requests 0 Set only the ERQ bit specified in the SERQ field 1 Set all bits in ERQ
5–3 Reserved	This field is reserved.
SERQ	Set Enable Request Sets the corresponding bit in ERQ.

25.3.13 Clear DONE Status Bit Register (DMA_CDNE)

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Ch offset = 4000_801Ch

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

DMA_CDNE field descriptions

Field	Description
7 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 CADN	Clears All DONE Bits 0 Clears only the TCDn_CSR[DONE] bit specified in the CDNE field 1 Clears all bits in TCDn_CSR[DONE]
5–3 Reserved	This field is reserved.
CDNE	Clear DONE Bit Clears the corresponding bit in TCDn_CSR[DONE]

25.3.14 Set START Bit Register (DMA_SSRT)

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Dh offset = 4000_801Dh

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

DMA_SSRT field descriptions

Field	Description
7 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 SAST	Set All START Bits (activates all channels) 0 Set only the TCDn_CSR[START] bit specified in the SSRT field 1 Set all bits in TCDn_CSR[START]
5–3 Reserved	This field is reserved.
SSRT	Set START Bit Sets the corresponding bit in TCDn_CSR[START]

25.3.15 Clear Error Register (DMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Eh offset = 4000_801Eh

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

DMA_CERR field descriptions

Field	Description
7 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 CAEI	Clear All Error Indicators 0 Clear only the ERR bit specified in the CERR field 1 Clear all bits in ERR
5–3 Reserved	This field is reserved.
CERR	Clear Error Indicator Clears the corresponding bit in ERR

25.3.16 Clear Interrupt Request Register (DMA_CINT)

The CINT provides a simple, memory-mapped mechanism to clear a given bit in the INT to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 4000_8000h base + 1Fh offset = 4000_801Fh

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

DMA_CINT field descriptions

Field	Description
7 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
6 CAIR	Clear All Interrupt Requests 0 Clear only the INT bit specified in the CINT field 1 Clear all bits in INT
5–3 Reserved	This field is reserved.
CINT	Clear Interrupt Request Clears the corresponding bit in INT

25.3.17 Interrupt Request Register (DMA_INT)

The INT register provides a bit map for the 8 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

Address: 4000_8000h base + 24h offset = 4000_8024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_INT field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 INT7	Interrupt Request 7 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT6	Interrupt Request 6 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT5	Interrupt Request 5

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT4	Interrupt Request 4 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT3	Interrupt Request 3 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT2	Interrupt Request 2 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT1	Interrupt Request 1 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
0 INT0	Interrupt Request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

25.3.18 Error Register (DMA_ERR)

The ERR provides a bit map for the channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, and then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Memory map/register definition

Address: 4000_8000h base + 2Ch offset = 4000_802Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERR field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ERR7	Error In Channel 7 0 An error in this channel has not occurred 1 An error in this channel has occurred
6 ERR6	Error In Channel 6 0 An error in this channel has not occurred 1 An error in this channel has occurred
5 ERR5	Error In Channel 5 0 An error in this channel has not occurred 1 An error in this channel has occurred
4 ERR4	Error In Channel 4 0 An error in this channel has not occurred 1 An error in this channel has occurred
3 ERR3	Error In Channel 3 0 An error in this channel has not occurred 1 An error in this channel has occurred
2 ERR2	Error In Channel 2 0 An error in this channel has not occurred 1 An error in this channel has occurred
1 ERR1	Error In Channel 1 0 An error in this channel has not occurred 1 An error in this channel has occurred
0 ERR0	Error In Channel 0 0 An error in this channel has not occurred 1 An error in this channel has occurred

25.3.19 Hardware Request Status Register (DMA_HRS)

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

Address: 4000_8000h base + 34h offset = 4000_8034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								HRS7	HRS6	HRS5	HRS4	HRS3	HRS2	HRS1	HRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_HRS field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HRS7	Hardware Request Status Channel 7 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 7 is not present 1 A hardware service request for channel 7 is present
6 HRS6	Hardware Request Status Channel 6 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 6 is not present 1 A hardware service request for channel 6 is present
5 HRS5	Hardware Request Status Channel 5

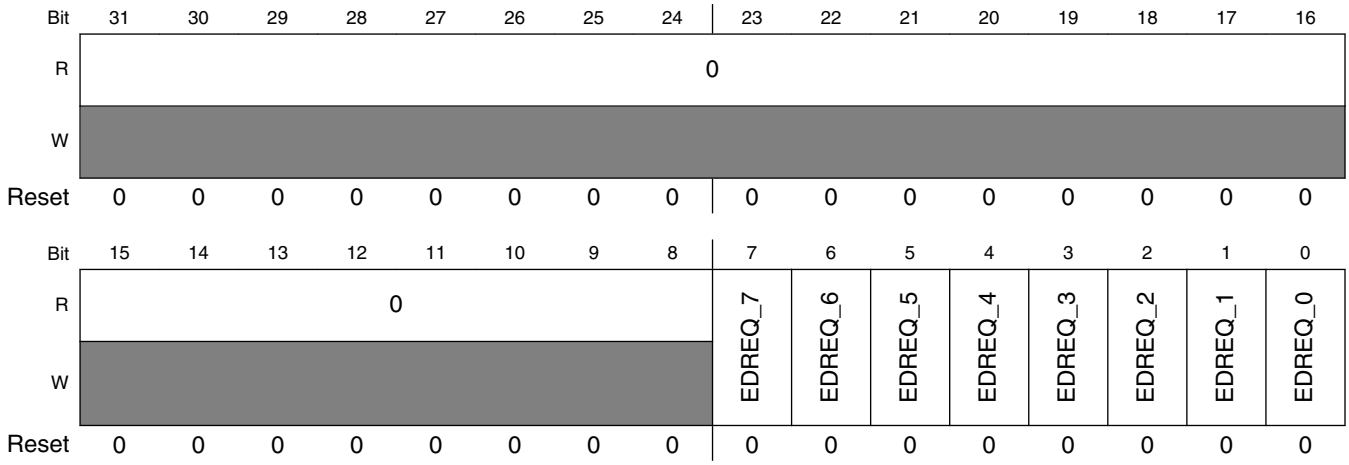
Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 5 is not present 1 A hardware service request for channel 5 is present
4 HRS4	Hardware Request Status Channel 4 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 4 is not present 1 A hardware service request for channel 4 is present
3 HRS3	Hardware Request Status Channel 3 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 3 is not present 1 A hardware service request for channel 3 is present
2 HRS2	Hardware Request Status Channel 2 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 2 is not present 1 A hardware service request for channel 2 is present
1 HRS1	Hardware Request Status Channel 1 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 1 is not present 1 A hardware service request for channel 1 is present
0 HRS0	Hardware Request Status Channel 0 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 0 is not present 1 A hardware service request for channel 0 is present

25.3.20 Enable Asynchronous Request in Stop Register (DMA_EARS)

Address: 4000_8000h base + 44h offset = 4000_8044h



DMA_EARS field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 EDREQ_7	Enable asynchronous DMA request in stop mode for channel 7 0 Disable asynchronous DMA request for channel 7. 1 Enable asynchronous DMA request for channel 7.
6 EDREQ_6	Enable asynchronous DMA request in stop mode for channel 6 0 Disable asynchronous DMA request for channel 6. 1 Enable asynchronous DMA request for channel 6.
5 EDREQ_5	Enable asynchronous DMA request in stop mode for channel 5 0 Disable asynchronous DMA request for channel 5. 1 Enable asynchronous DMA request for channel 5.
4 EDREQ_4	Enable asynchronous DMA request in stop mode for channel 4 0 Disable asynchronous DMA request for channel 4. 1 Enable asynchronous DMA request for channel 4.
3 EDREQ_3	Enable asynchronous DMA request in stop mode for channel 3. 0 Disable asynchronous DMA request for channel 3. 1 Enable asynchronous DMA request for channel 3.
2 EDREQ_2	Enable asynchronous DMA request in stop mode for channel 2. 0 Disable asynchronous DMA request for channel 2. 1 Enable asynchronous DMA request for channel 2.
1 EDREQ_1	Enable asynchronous DMA request in stop mode for channel 1.

Table continues on the next page...

DMA_EARS field descriptions (continued)

Field	Description
	0 Disable asynchronous DMA request for channel 1 1 Enable asynchronous DMA request for channel 1.
0 EDREQ_0	Enable asynchronous DMA request in stop mode for channel 0. 0 Disable asynchronous DMA request for channel 0. 1 Enable asynchronous DMA request for channel 0.

25.3.21 Channel n Priority Register (DMA_DCHPRI*n*)

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 7.

Address: 4000_8000h base + 100h offset + (1d × i), where i=0d to 7d

Bit	7	6	5	4	3	2	1	0
Read	ECP	DPA	0			CHPRI		
Write								
Reset	0	0	0	0	0	*	*	*

* Notes:
- CHPRI field: See bit field description.

DMA_DCHPRI*n* field descriptions

Field	Description
7 ECP	Enable Channel Preemption. 0 Channel n cannot be suspended by a higher priority channel's service request. 1 Channel n can be temporarily suspended by the service request of a higher priority channel.
6 DPA	Disable Preempt Ability. 0 Channel n can suspend a lower priority channel. 1 Channel n cannot suspend any channel, regardless of channel priority.
5–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled **NOTE:** Reset value for the channel priority field, CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI7[CHPRI] = 0b111.

25.3.22 TCD Source Address (DMA_TCD*n*_SADDR)

Address: 4000_8000h base + 1000h offset + (32d × 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	SADDR																															
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.

DMA_TCD*n*_SADDR field descriptions

Field	Description
SADDR	Source Address Memory address pointing to the source data.

25.3.23 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

Address: 4000_8000h base + 1004h offset + (32d × i), where i=0d to 7d

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

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_SOFF field descriptions

Field	Description
SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

25.3.24 TCD Transfer Attributes (DMA_TCD*n*_ATTR)

Address: 4000_8000h base + 1006h offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	SMOD					SSIZE			DMOD					DSIZE		
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_ATTR field descriptions

Field	Description
15–11 SMOD	Source Address Modulo 0 Source address modulo feature is disabled ≠0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
10–8 SSIZE	Source data transfer size **NOTE:** Using a Reserved value causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 Reserved 100 16-byte 101 32-byte 110 Reserved 111 Reserved
7–3 DMOD	Destination Address Modulo See the SMOD definition
DSIZE	Destination data transfer size See the SSIZE definition

25.3.25 TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD*n*_NBYTES_MLNO)

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is disabled (CR[EMLM] = 0)

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

Address: 4000_8000h base + 1008h offset + (32d × 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	NBYTES																															
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.

DMA_TCD*n*_NBYTES_MLNO field descriptions

Field	Description
NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

25.3.26 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)

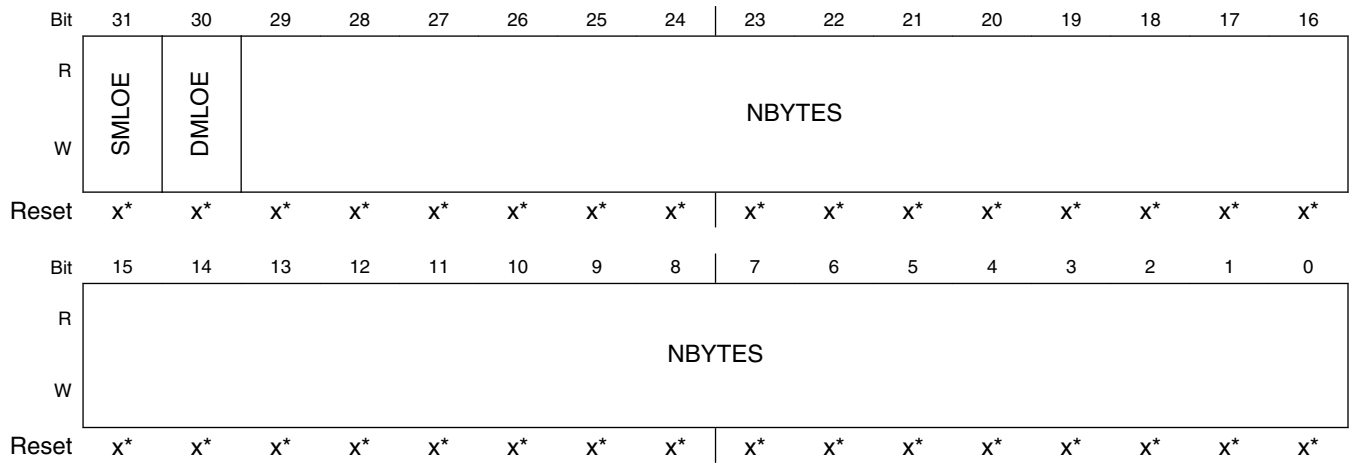
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 4000_8000h base + 1008h offset + (32d × i), where i=0d to 7d



* Notes:
- x = Undefined at reset.

DMA_TCD*n*_NBYTES_MLOFFNO field descriptions

Field	Description
31 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

25.3.27 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)

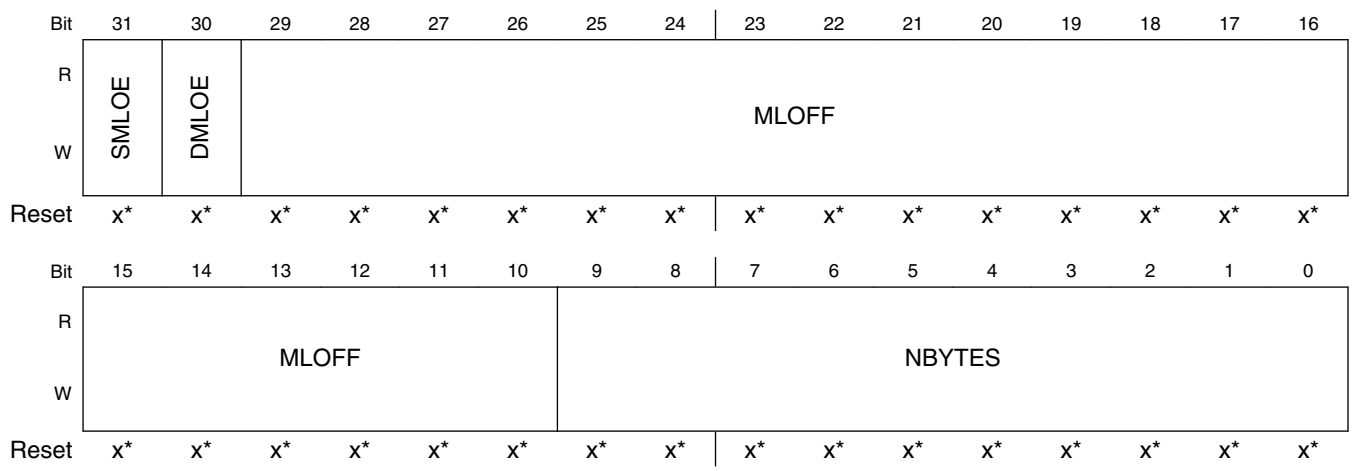
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 4000_8000h base + 1008h offset + (32d × i), where i=0d to 7d



* Notes:
- x = Undefined at reset.

DMA_TCD*n*_NBYTES_MLOFFYES field descriptions

Field	Description
31 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion.

Table continues on the next page...

DMA_TCD*n*_NBYTES_MLOFFYES field descriptions (continued)

Field	Description
	0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
29–10 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

25.3.28 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Address: 4000_8000h base + 100Ch offset + (32d × 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	SLAST																															
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.

DMA_TCD*n*_SLAST field descriptions

Field	Description
SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure. This register uses two's complement notation; the overflow bit is discarded.

25.3.29 TCD Destination Address (DMA_TCD*n*_DADDR)

Address: 4000_8000h base + 1010h offset + (32d × 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	DADDR																															
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.

DMA_TCD*n*_DADDR field descriptions

Field	Description
DADDR	Destination Address Memory address pointing to the destination data.

25.3.30 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

Address: 4000_8000h base + 1014h offset + (32d × i), where i=0d to 7d

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

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_DOFF field descriptions

Field	Description
DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

25.3.31 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)

If TCDn_CITER[ELINK] is set, the TCDn_CITER register is defined as follows.

Address: 4000_8000h base + 1016h offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8
Read Write	ELINK	0			LINKCH			CITER
Reset	x*	x*	x*	x*	x*	x*	x*	x*

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

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_CITER_ELINKYES field descriptions

Field	Description
15 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
14–12 Reserved	This field is reserved.
11–9 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

25.3.32 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

If TCDn_CITER[ELINK] is cleared, the TCDn_CITER register is defined as follows.

Address: 4000_8000h base + 1016h offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8
Read Write	ELINK	CITER						
Reset	x*	x*	x*	x*	x*	x*	x*	x*

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

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_CITER_ELINKNO field descriptions

Field	Description
15 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

25.3.33 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Address: 4000_8000h base + 1018h offset + (32d × 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	DLASTSGA																															
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.

DMA_TCD*n*_DLASTSGA field descriptions

Field	Description
DLASTSGA	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: • Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, otherwise a configuration error is reported.

25.3.34 TCD Control and Status (DMA_TCD*n*_CSR)

Address: 4000_8000h base + 101Ch offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8
Read	BWC					MAJORLINKCH		
Write			0					
Reset	x*	x*	x*	x*	x*	x*	x*	x*

Bit	7	6	5	4	3	2	1	0
Read	DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Write								
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_CSR field descriptions

Field	Description
15–14 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch. 00 No eDMA engine stalls. 01 Reserved 10 eDMA engine stalls for 4 cycles after each R/W. 11 eDMA engine stalls for 8 cycles after each R/W.
13–11 Reserved	This field is reserved.
10–8 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
7 DONE	Channel Done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software clears it, or the hardware when the channel is activated. The access of this field is W0C, write-zero-to-clear. **NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits.
6 ACTIVE	Channel Active This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected.
5 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
4 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set.

Table continues on the next page...

DMA_TCD*n*_CSR field descriptions (continued)

Field	Description
	0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
3 DREQ	Disable Request If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero. 0 The channel's ERQ bit is not affected. 1 The channel's ERQ bit is cleared when the major loop is complete.
2 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
1 INTMAJOR	Enable an interrupt when major iteration count completes. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
0 START	Channel Start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

25.3.35 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

If the TCDn_BITER[ELINK] bit is set, the TCDn_BITER register is defined as follows.

Address: 4000_8000h base + 101Eh offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8
Read Write	ELINK	0			LINKCH			BITER
Reset	x*	x*	x*	x*	x*	x*	x*	x*

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

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_BITER_ELINKYES field descriptions

Field	Description
15 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
14–12 Reserved	This field is reserved.
11–9 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.
BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.

Table continues on the next page...

DMA_TCD*n*_BITER_ELINKYES field descriptions (continued)

Field	Description
	NOTE: When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

25.3.36 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

If the TCDn_BITER[ELINK] bit is cleared, the TCDn_BITER register is defined as follows.

Address: 4000_8000h base + 101Eh offset + (32d × i), where i=0d to 7d

Bit	15	14	13	12	11	10	9	8
Read Write	ELINK	BITER						
Reset	x*	x*	x*	x*	x*	x*	x*	x*

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

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_BITER_ELINKNO field descriptions

Field	Description
15 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the

Table continues on the next page...

DMA_TCD*n*_BITER_ELINKNO field descriptions (continued)

Field	Description
	contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

25.4 Functional description

The operation of the eDMA is described in the following subsections.

25.4.1 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

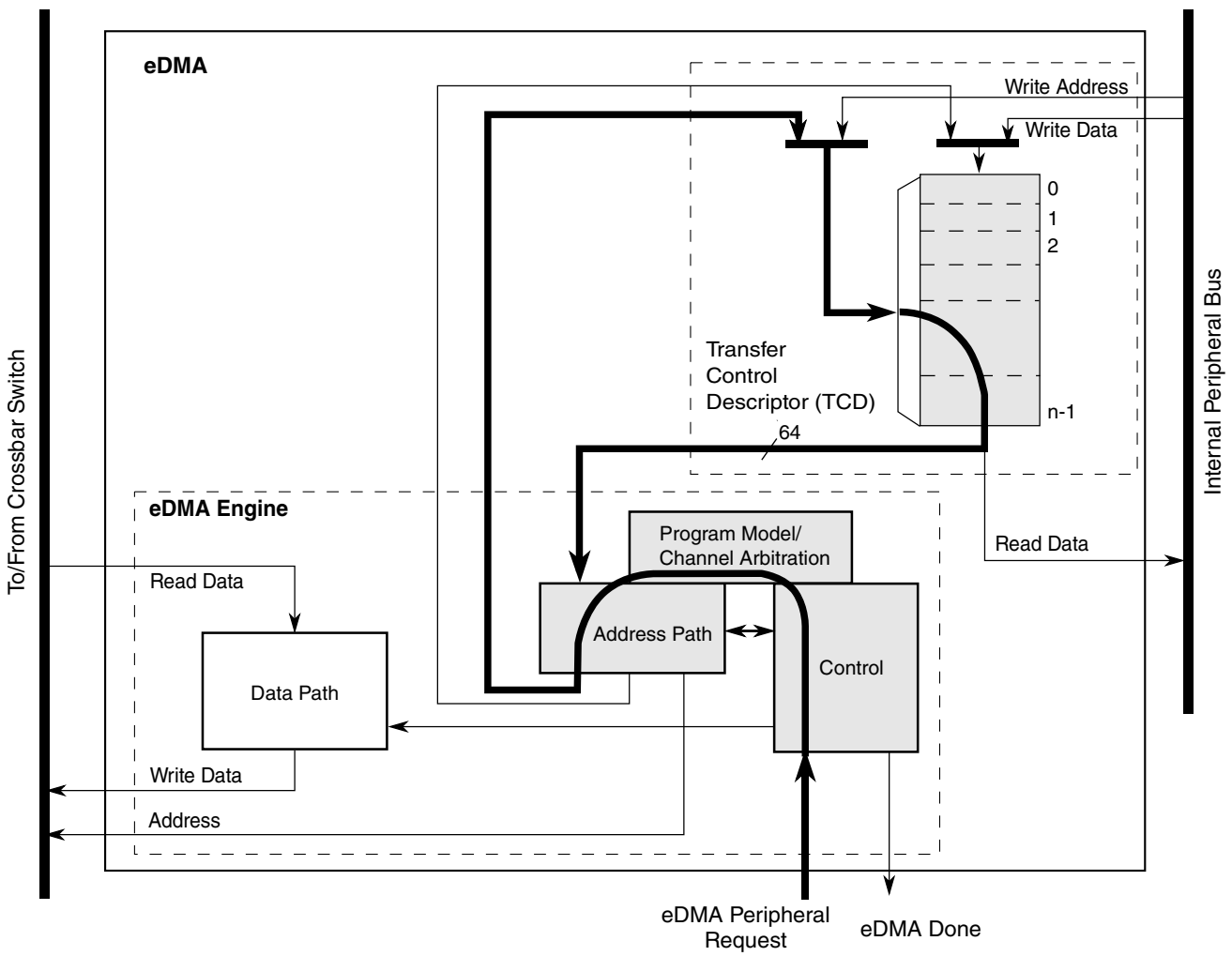


Figure 25-2. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration performs, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

The following diagram illustrates the second part of the basic data flow:

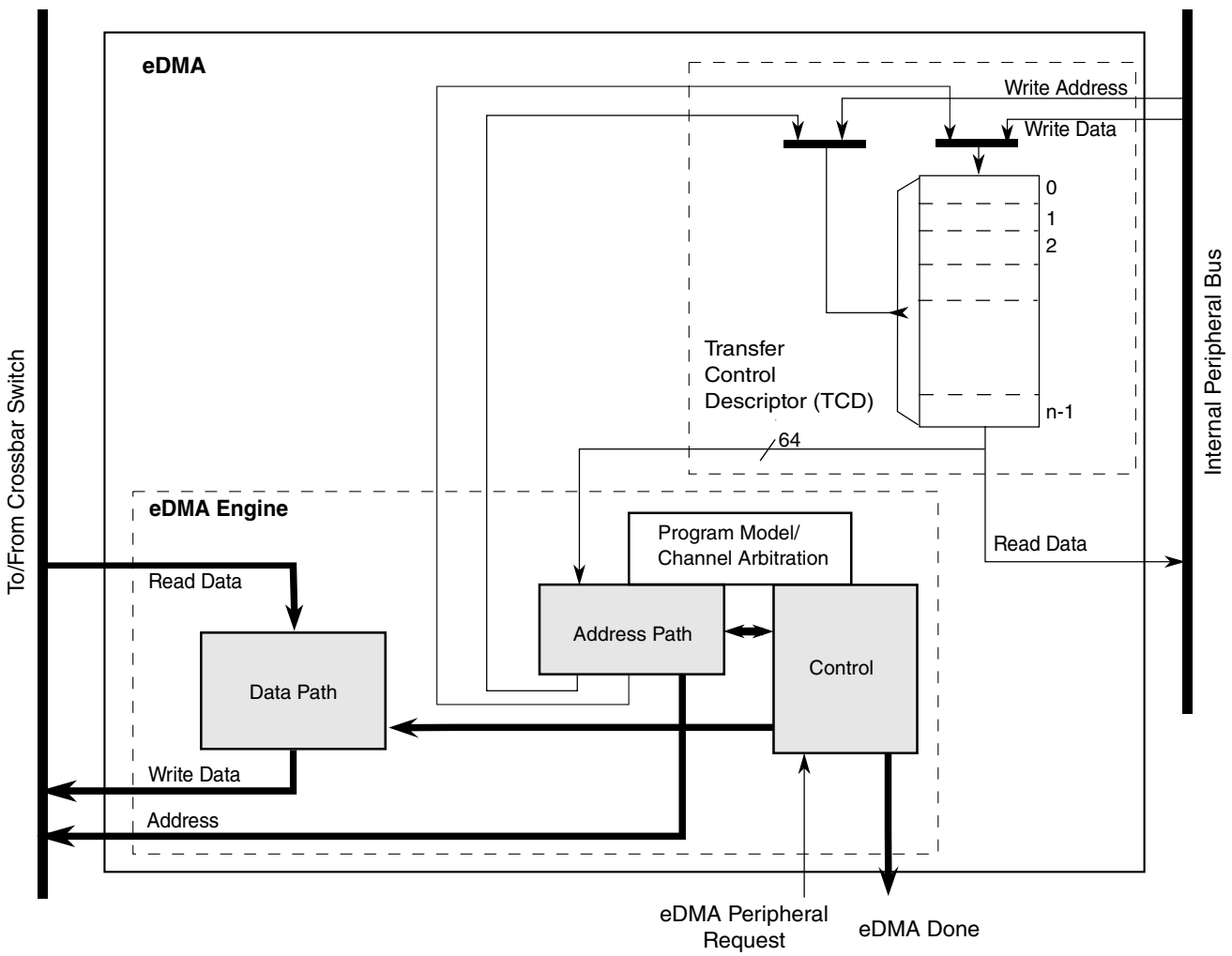


Figure 25-3. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

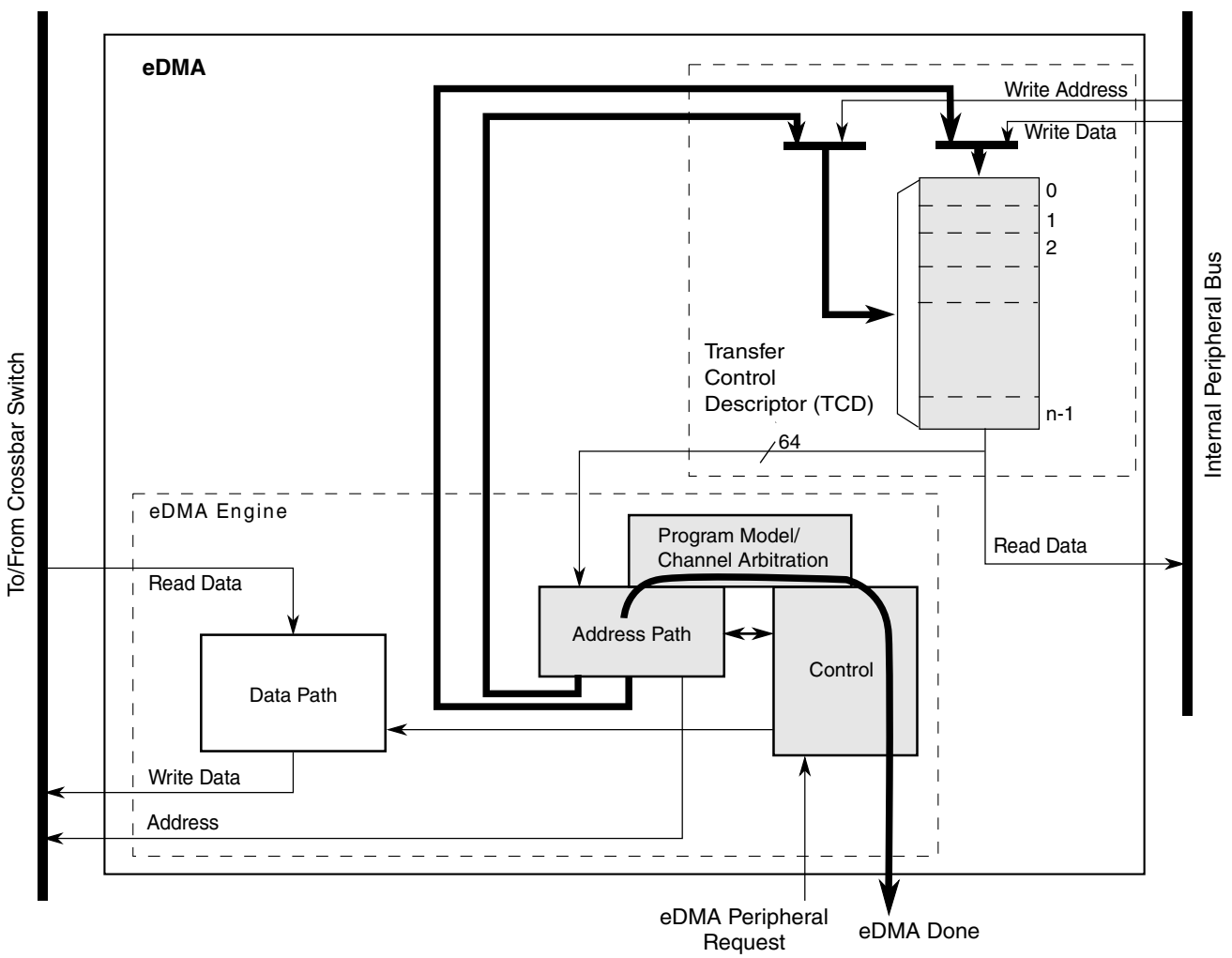


Figure 25-4. eDMA operation, part 3

25.4.2 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.

NOTE

When two channels have the same priority, a channel priority error exists and will be reported in the Error Status register. However, the channel number will not be reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration is undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control Register. If all of the channel priorities within a group are not unique, the DMA will be halted after the CPE error is recorded. The DMA will remain halted and will not process any channel service requests. Once all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the Halt bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

NOTE

The cancel transfer request allows the user to stop a large data transfer in the event the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application

software must handle the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor since the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

25.4.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

25.4.4 Performance

This section addresses the performance of the eDMA module, focusing on two separate metrics:

- In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces.
- In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more relevant metric. In this environment, the speed of the source and destination address spaces remains important. However, the microarchitecture of the eDMA also factors significantly into the resulting metric.

25.4.4.1 Peak transfer rates

The peak transfer rates for several different source and destination transfers are shown in the following tables. These tables assume:

- Internal SRAM can be accessed with zero wait-states when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states, when viewed from the system bus data phase
- All internal peripheral bus accesses are 32-bits in size

NOTE

All architectures will not meet the assumptions listed above. See the SRAM configuration section for more information.

This table compares peak transfer rates based on different possible system speeds. Specific chips/devices may not support all system speeds listed.

Table 25-4. eDMA peak transfer rates (Mbytes/sec)

System Speed, Width	Internal SRAM-to-Internal SRAM	32 bit internal peripheral bus-to-Internal SRAM	Internal SRAM-to-32 bit internal peripheral bus
48 MHz, 32 bit	96.0	48.0	38.4
83.3 MHz, 32 bit	166.7	83.3	66.7
100.0 MHz, 32 bit	200.0	100.0	80.0
133.3 MHz, 32 bit	266.7	133.3	106.7
150.0 MHz, 32 bit	300.0	150.0	120.0

Internal-SRAM-to-internal-SRAM transfers occur at the core's datapath width. For all transfers involving the internal peripheral bus, 32-bit transfer sizes are used. In all cases, the transfer rate includes the time to read the source plus the time to write the destination.

25.4.4.2 Peak request rates

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, assume that the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel.

The eDMA design supports the following hardware service request sequence. Note that the exact timing from Cycle 7 is a function of the response times for the channel's read and write accesses. In the case of an internal peripheral bus read and internal SRAM write, the combined data phase time is 4 cycles. For an SRAM read and internal peripheral bus write, it is 5 cycles.

Table 25-5. Hardware service request process

Cycle		Description
With internal peripheral bus read and internal SRAM write	With SRAM read and internal peripheral bus write	
1		eDMA peripheral request is asserted.
2		The eDMA peripheral request is registered locally in the eDMA module and qualified. TCD*n*_CSR[START] bit initiated requests start at this point with the registering of the user write to TCD*n* word 7.
3		Channel arbitration begins.
4		Channel arbitration completes. The transfer control descriptor local memory read is initiated.
5–6		The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles
7		The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
8–11	8–12	The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.
12	13	This cycle represents the data phase of the last destination write.

Table continues on the next page...

Table 25-5. Hardware service request process (continued)

Cycle		Description
With internal peripheral bus read and internal SRAM write	With SRAM read and internal peripheral bus write	
13	14	The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. The TCD*n* word 7 is read and checked for channel linking or scatter/gather requests.
14	15	The appropriate fields in the first part of the TCD*n* are written back into the local memory.
15	16	The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
16	17	The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the system bus, DMA requests can be processed every 9 cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (4 cycles) and SRAM-to-internal peripheral bus (5 cycles), DMA requests can be processed every 11.5 cycles (4 + (4+5)/2 + 3). This is the time from Cycle 4 to Cycle x +5. The resulting peak request rate, as a function of the system frequency, is shown in the following table.

Table 25-6. eDMA peak request rate (MReq/sec)

System frequency (MHz)	Request rate with zero wait states	Request rate with wait states
48.0	5.3	4.2
66.6	7.4	5.8
83.3	9.2	7.2
100.0	11.1	8.7
133.3	14.8	11.6
150.0	16.6	13.0

A general formula to compute the peak request rate with overlapping requests is:

PEAKreq = freq / [entry + (1 + read_ws) + (1 + write_ws) + exit]

where:

Table 25-7. Peak request formula operands

Operand	Description
PEAKreq	Peak request rate
freq	System frequency
entry	Channel startup (4 cycles)
read_ws	Wait states seen during the system bus read data phase
write_ws	Wait states seen during the system bus write data phase
exit	Channel shutdown (3 cycles)

25.4.4.3 eDMA performance example

Consider a system with the following characteristics:

- Internal SRAM can be accessed with one wait-state when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states viewed from the system bus data phase
- System operates at 150 MHz

For an SRAM to internal peripheral bus transfer,

PEAKreq = 150 MHz / [4 + (1 + 1) + (1 + 3) + 3] cycles = 11.5 Mreq/sec

For an internal peripheral bus to SRAM transfer,

PEAKreq = 150 MHz / [4 + (1 + 2) + (1 + 1) + 3] cycles = 12.5 Mreq/sec

Assuming an even distribution of the two transfer types, the average peak request rate would be:

PEAKreq = (11.5 Mreq/sec + 12.5 Mreq/sec) / 2 = 12.0 Mreq/sec

The minimum number of cycles to perform a single read/write, zero wait states on the system bus, from a cold start where no channel is executing and eDMA is idle are:

- 11 cycles for a software, that is, a TCD*n*_CSR[START] bit, request
- 12 cycles for a hardware, that is, an eDMA peripheral request signal, request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering is absorbed in or overlaps the previous executing channel.

Note

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

25.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

25.5.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.
2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI register if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service via either:
   - Software: setting the TCD*n*_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in the following table, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 25-8. TCD Control and Status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (Automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

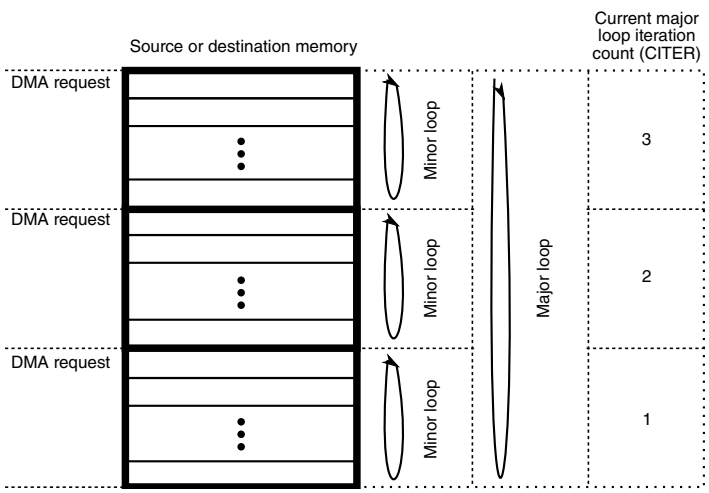


Figure 25-5. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

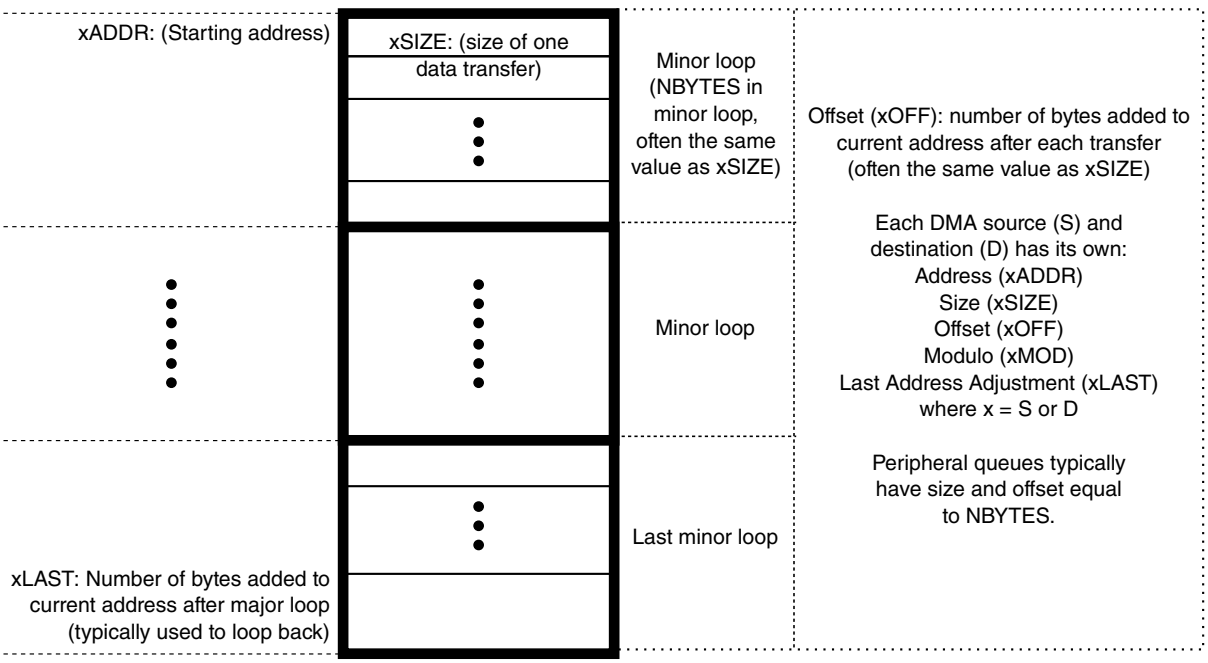


Figure 25-6. Memory array terms

25.5.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than channel priority error, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

25.5.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

25.5.3.1 Fixed channel arbitration

In this mode, the channel service request from the highest priority channel is selected to execute.

25.5.3.2 Round-robin channel arbitration

Channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels.

25.5.4 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

25.5.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32-bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32-bits to location 0x200C → last iteration of the minor loop → major loop complete.
6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

25.5.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, eDMA peripheral, request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
    a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
    b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
    c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
    d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
    e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
    f. Write 32-bits to location 0x2008 → third iteration of the minor loop.
    g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
    h. Write 32-bits to location 0x200C → last iteration of the minor loop.
6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.

8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:
    a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
    b. Write 32-bits to location 0x2010 → first iteration of the minor loop.
    c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
    d. Write 32-bits to location 0x2014 → second iteration of the minor loop.
    e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
    f. Write 32-bits to location 0x2018 → third iteration of the minor loop.
    g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
    h. Write 32-bits to location 0x201C → last iteration of the minor loop → major loop complete.
14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

25.5.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes and the MOD field is set to 4, allowing for a 2^4 byte (16-byte) size queue.

Table 25-9. Modulo example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

25.5.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

25.5.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware, that is, peripheral, initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

25.5.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

25.5.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected as the channel arbitration mode. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

25.5.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done→ set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a nine bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

The TCD*n*_CITER[E_LINK] bit and the TCD*n*_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, i.e, use another channel's TCD, at the end of a loop.

Table 25-10. Channel Linking Parameters

Desired Link Behavior	TCD Control Field Name	Description
Link at end of Minor Loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of Major Loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

25.5.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

25.5.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode,
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

25.5.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution (see the diagram in TCD structure). This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write 1 to the TCD.major.e_link bit.
2. Read back the TCD.major.e_link bit.
3. Test the TCD.major.e_link request status:
   - If TCD.major.e_link = 1, the dynamic link attempt was successful.
   - If TCD.major.e_link = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

25.5.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources.When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

25.5.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD indentification (ID).

1. When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2. Write 1b to the TCD.d_req bit.

   Should a dynamic scatter/gather attempt fail, setting the TCD.d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

3. Write the TCD.dlast_sga field with the scatter/gather address.
4. Write 1b to the TCD.e_sg bit.
5. Read back the 16 bit TCD control/status field.

6. Test the TCD.e_sg request status and TCD.major.linkch value:

   If e_sg = 1b, the dynamic link attempt was successful.

   If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

25.5.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD indentification (ID).

1. Write 1b to the TCD.d_req bit.

   Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

2. Write theTCD.dlast_sga field with the scatter/gather address.
3. Write 1b to the TCD.e_sg bit.
4. Read back the TCD.e_sg bit.
5. Test the TCD.e_sg request status:

   If e_sg = 1b, the dynamic link attempt was successful.

   If e_sg = 0b, read the 32 bit TCD dlast_sga field.

   If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

Chapter 26
External Watchdog Monitor (EWM)

26.1 Introduction

NOTE

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

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 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.

The EWM differs from the internal watchdog in that it does not reset the MCU's CPU and peripherals. The EWM provides an independent $\overline{\text{EWM\_out}}$ signal that when asserted resets or places an external circuit into a safe mode. The $\overline{\text{EWM\_out}}$ signal is asserted upon the EWM counter time-out. An optional external input EWM_in is provided to allow additional control of the assertion of $\overline{\text{EWM\_out}}$ signal.

26.1.1 Features

Features of EWM module include:

- Independent LPO_CLK clock source
- Programmable time-out period specified in terms of number of EWM LPO_CLK 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_refresh_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 assertion of the $\overline{\text{EWM\_out}}$ signal.

26.1.2 Modes of Operation

This section describes the module's operating modes.

26.1.2.1 Stop Mode

When the EWM is in stop mode, the CPU refreshes 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 refresh mechanism: At the exit from stop mode by an interrupt, refresh mechanism state machine starts from the previous state which means, if first refresh command is written correctly and EWM enters the stop mode immediately, the next command has to be written within the next 15 (*EWM_refresh_time*) peripheral bus clocks after exiting from stop mode. User must mask all interrupts prior to executing EWM refresh instructions.

26.1.2.2 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.

26.1.3 Block Diagram

This figure shows the EWM block diagram.

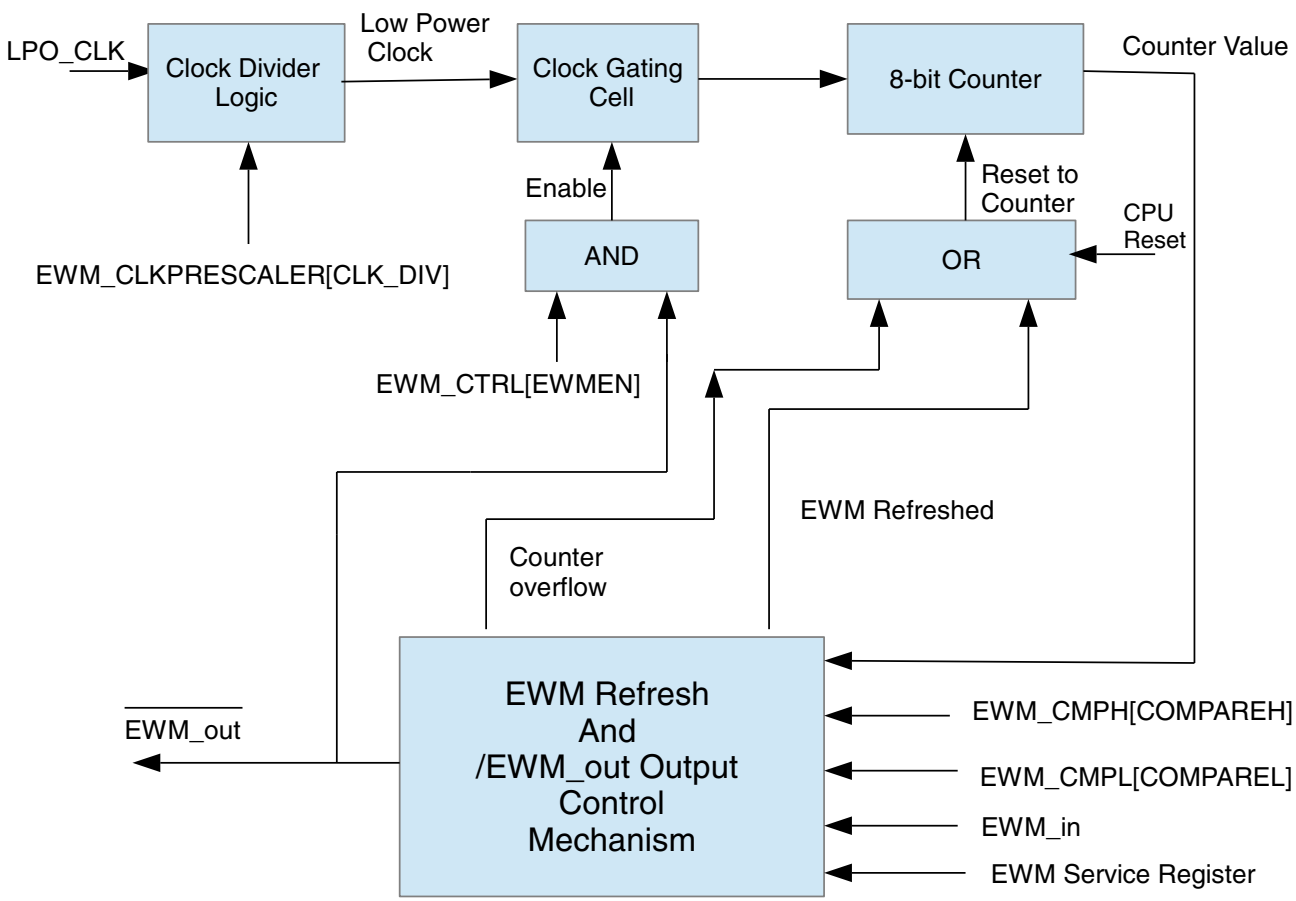


Figure 26-1. EWM Block Diagram

26.2 EWM Signal Descriptions

The EWM has two external signals and internal options for the counter clock sources, as shown in the following table.

Table 26-1. 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

26.3 Memory Map/Register Definition

This section contains the module memory map and registers.

EWM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_1000	Control Register (EWM_CTRL)	8	R/W	00h	26.3.1/516
4006_1001	Service Register (EWM_SERV)	8	W (always reads 0)	00h	26.3.2/517
4006_1002	Compare Low Register (EWM_CMPL)	8	R/W	00h	26.3.3/517
4006_1003	Compare High Register (EWM_CMPH)	8	R/W	FFh	26.3.4/518
4006_1004	Clock Control Register (EWM_CLKCTRL)	8	R/W	00h	26.3.5/519
4006_1005	Clock Prescaler Register (EWM_CLKPRESCALER)	8	R/W	00h	26.3.6/519

26.3.1 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: 4006_1000h base + 0h offset = 4006_1000h

Bit	7	6	5	4	3	2	1	0
Read	0				INTEN	INEN	ASSIN	EWMEN
Write								
Reset	0	0	0	0	0	0	0	0

EWM_CTRL field descriptions

Field	Description
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 the ASSIN bit inverts the assert state of EWM_in signal 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. This bit when unset, keeps the EWM module disabled. It cannot be re-enabled until a next reset, due to the write-once nature of this bit.

26.3.2 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: 4006_1000h base + 1h offset = 4006_1001h

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

EWM_SERV field descriptions

Field	Description
SERVICE	The EWM refresh 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 refresh is invalid 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_refresh_time*.

26.3.3 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 refresh 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: 4006_1000h base + 2h offset = 4006_1002h

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

EWM_CMPL field descriptions

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 refresh time is required.

26.3.4 Compare High Register (EWM_CMPH)

The CMPH register is reset to 0xFF after a CPU reset. This provides a maximum of 256 clocks time, for the CPU to refresh 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 0xFE because the EWM counter never expires when CMPH = 0xFF. The expiration happens only if EWM counter is greater than CMPH.

Address: 4006_1000h base + 3h offset = 4006_1003h

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

EWM_CMPH field descriptions

Field	Description
COMPAREH	To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) maximum refresh time is required.

26.3.5 Clock Control Register (EWM_CLKCTRL)

This CLKCTRL register is reset to 0x00 after a CPU reset.

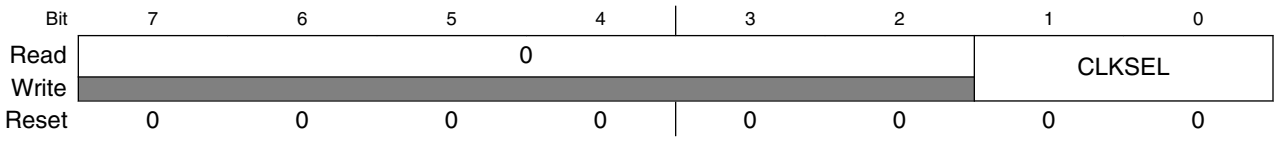
NOTE

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: 4006_1000h base + 4h offset = 4006_1004h



EWM_CLKCTRL field descriptions

Field	Description
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.

26.3.6 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-specific information for details.

Address: 4006_1000h base + 5h offset = 4006_1005h

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

EWM_CLKPRESCALER field descriptions

Field	Description
CLK_DIV	Selected low power clock source for running the EWM counter can be prescaled as below. • Prescaled clock frequency = low power clock source frequency / (1 + CLK_DIV)

26.4 Functional Description

The following sections describe functional details of the EWM module.

NOTE

When the BUS_CLK is lost, then EWM module doesn't generate the $\overline{\text{EWM\_out}}$ signal and no refresh operation is possible

26.4.1 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 refreshed by the CPU within the programmable refresh window, indicating that the application code is executed as expected.

The $\overline{\text{EWM\_out}}$ signal is asserted in any of the following conditions:

- The EWM refresh occurs when the counter value is less than CMPL value.
- The EWM counter value reaches the CMPH value, and no EWM refresh has occurred.
- If functionality of EWM_in pin is enabled and EWM_in pin is asserted while refreshing the EWM.
- After any reset (by the virtue of the external pull-down mechanism on the $\overline{\text{EWM\_out}}$ pin)

The $\overline{\text{EWM\_out}}$ is asserted after any reset by the virtue of the external pull-down mechanism on the $\overline{\text{EWM\_out}}$ signal. Then, to deassert the $\overline{\text{EWM\_out}}$ signal, 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. The pad state is controlled by the $\overline{\text{EWM\_out}}$ signal only after the EWM is enabled by the EWMEN bit in the CTRL register.

Note

$\overline{\text{EWM\_out}}$ pad must be in pull down state when EWM functionality is used and when EWM is under Reset.

26.4.2 The EWM_in Signal

The EWM_in is a digital input signal for safety status of external safety circuits, that allows an external circuit to control the assertion of the $\overline{\text{EWM\_out}}$ signal. For example, in the application, an external circuit monitors a critical safety function, and if there is fault with safety function, the external circuit can then actively initiate the $\overline{\text{EWM\_out}}$ 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 start refreshing the EWM. This ensures that the $\overline{\text{EWM\_out}}$ stays in the deasserted state; otherwise, the $\overline{\text{EWM\_out}}$ output signal is asserted.

Note

The user must update the CMPH and CMPL registers prior to enabling the EWM. After enabling the EWM, the counter resets to zero, therefore the user shall provide a reasonable time after a power-on reset for the external monitoring circuit to stabilize. The user shall also ensure that the EWM_in pin is deasserted.

26.4.3 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 the CPU reset, or when EWM refresh action completes, or at counter overflow. The counter value is not accessible to the CPU.

26.4.4 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 refresh window to refresh the EWM module.

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.

26.4.5 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 for correct EWM refresh operation. Therefore, three possible conditions can occur:

Table 26-2. EWM Refresh Mechanisms

Condition	Mechanism
An EWM refresh action completes when: CMPL < Counter < CMPH.	The software behaves as expected and the EWM counter is reset to zero. The $\overline{\text{EWM\_out}}$ output signal remains in the deasserted state if, during the EWM refresh action, the EWM_in input has been in deasserted state..
An EWM refresh action completes when Counter < CMPL	The software refreshes the EWM before the windowed time frame, the counter is reset to zero and the $\overline{\text{EWM\_out}}$ output signal is asserted irrespective of the input $\overline{\text{EWM\_in}}$.
Counter value reaches CMPH prior to completion of EWM refresh action.	Software has not refreshed the EWM. The EWM counter is reset to zero and the $\overline{\text{EWM\_out}}$ output signal is asserted irrespective of the input $\overline{\text{EWM\_in}}$.

26.4.6 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.

26.4.7 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].

26.4.8 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 27
Watchdog Timer (WDOG)

27.1 Introduction

NOTE

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

The Watchdog Timer (WDOG) keeps a watch on the system functioning and resets it in case of its failure. Reasons for failure include run-away software code and the stoppage of the system clock that in a safety critical system can lead to serious consequences. In such cases, the watchdog brings the system into a safe state of operation. The watchdog monitors the operation of the system by expecting periodic communication from the software, generally known as servicing or refreshing the watchdog. If this periodic refreshing does not occur, the watchdog resets the system.

27.2 Features

The features of the Watchdog Timer (WDOG) include:

- Clock source input independent from CPU/bus clock. Choice between two clock sources:
  - Low-power oscillator (LPO)
  - External system clock
- Unlock sequence for allowing updates to write-once WDOG control/configuration bits.
- All WDOG control/configuration bits are writable once only within 256 bus clock cycles of being unlocked.

    - You need to always update these bits after unlocking within 256 bus clock cycles. Failure to update these bits resets the system.
- Programmable time-out period specified in terms of number of WDOG clock cycles.
- Ability to test WDOG timer and reset with a flag indicating watchdog test.
    - Quick test—Small time-out value programmed for quick test.
    - Byte test—Individual bytes of timer tested one at a time.
    - Read-only access to the WDOG timer—Allows dynamic check that WDOG timer is operational.

NOTE

Reading the watchdog timer counter while running the watchdog on the bus clock might not give the accurate counter value.

- Windowed refresh option
    - Provides robust check that program flow is faster than expected.
    - Programmable window.
    - Refresh outside window leads to reset.
- Robust refresh mechanism
    - Write values of 0xA602 and 0xB480 to WDOG Refresh Register within 20 bus clock cycles.
- Count of WDOG resets as they occur.
- Configurable interrupt on time-out to provide debug breadcrumbs. This is followed by a reset after 256 bus clock cycles.

27.3 Functional overview

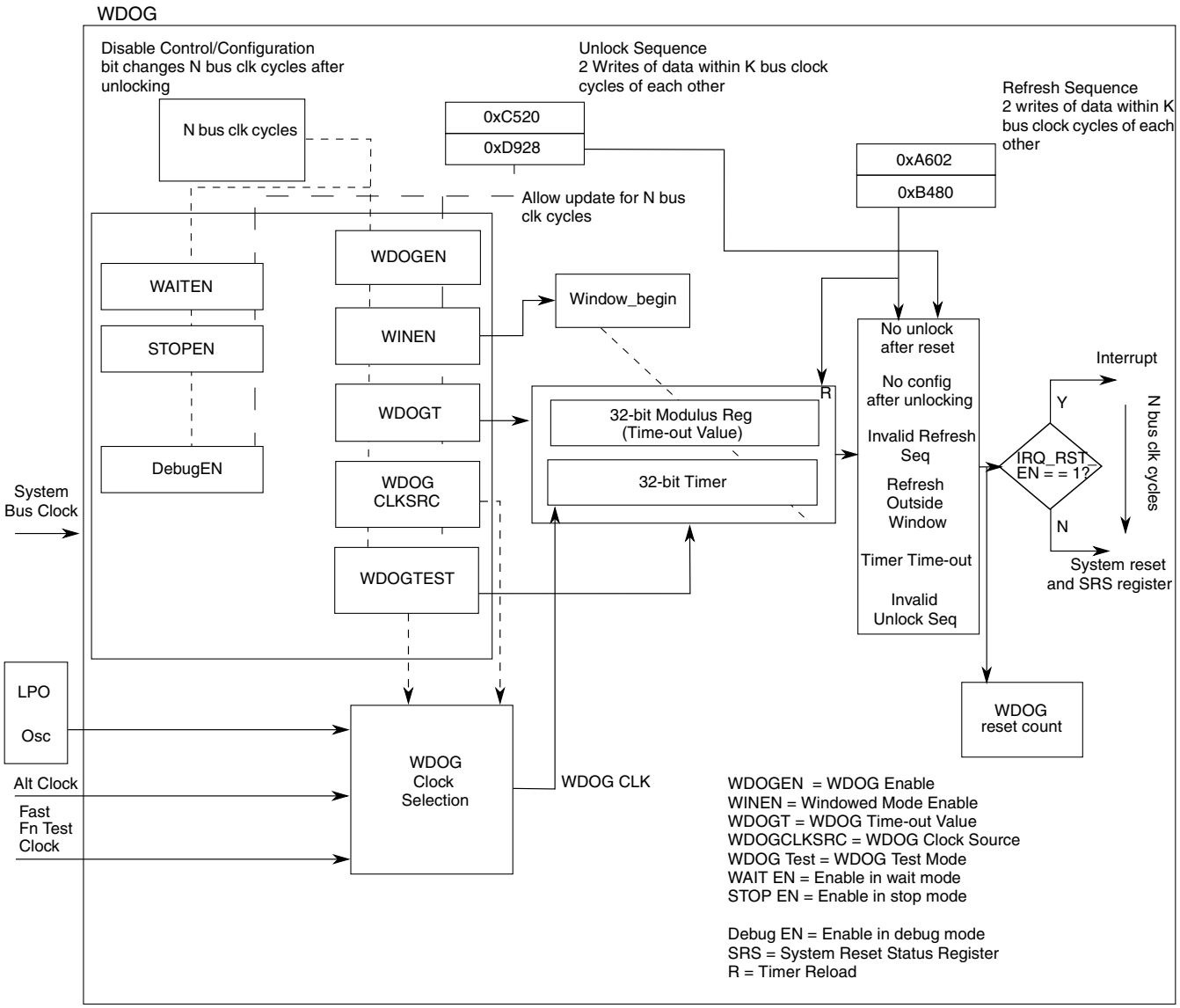


Figure 27-1. WDOG operation

The preceding figure shows the operation of the watchdog. The values for N and K are:

- N = 256
- K = 20

The watchdog is a fail safe mechanism that brings the system into a known initial state in case of its failure due to CPU clock stopping or a run-away condition in code execution. In its simplest form, the watchdog timer runs continuously off a clock source and expects to be serviced periodically, failing which it resets the system. This ensures that the software is executing correctly and has not run away in an unintended direction. Software can adjust the period of servicing or the time-out value for the watchdog timer to meet the needs of the application.

You can select a windowed mode of operation that expects the servicing to be done only in a particular window of the time-out period. An attempted servicing of the watchdog outside this window results in a reset. By operating in this mode, you can get an indication of whether the code is running faster than expected. The window length is also user programmable.

If a system fails to update/refresh the watchdog due to an unknown and persistent cause, it will be caught in an endless cycle of resets from the watchdog. To analyze the cause of such conditions, you can program the watchdog to first issue an interrupt, followed by a reset. In the interrupt service routine, the software can analyze the system stack to aid debugging.

To enhance the independence of watchdog from the system, it runs off an independent LPO oscillator clock. You can also switch over to an alternate clock source if required, through a control register bit.

27.3.1 Unlocking and updating the watchdog

As long as ALLOW_UPDATE in the watchdog control register is set, you can unlock and modify the write-once-only control and configuration registers:

1. Write 0xC520 followed by 0xD928 within 20 bus clock cycles to a specific unlock register (WDOG_UNLOCK).
2. Wait one bus clock cycle. You cannot update registers on the bus clock cycle immediately following the write of the unlock sequence.
3. An update window equal in length to the watchdog configuration time (WCT) opens. Within this window, you can update the configuration and control register bits.

These register bits can be modified only once after unlocking.

If none of the configuration and control registers is updated within the update window, the watchdog issues a reset, that is, interrupt-then-reset, to the system. Trying to unlock the watchdog within the WCT after an initial unlock has no effect. During the update operation, the watchdog timer is not paused and continues running in the background. After the update window closes, the watchdog timer restarts and the watchdog functions according to the new configuration.

The update feature is useful for applications that have an initial, non-safety critical part, where the watchdog is kept disabled or with a conveniently long time-out period. This means the application coder does not have to frequently service the watchdog. After the critical part of the application begins, the watchdog can be reconfigured as needed.

The watchdog issues a reset, that is, interrupt-then-reset if enabled, to the system for any of these invalid unlock sequences:

- Write any value other than 0xC520 or 0xD928 to the unlock register.
- ALLOW_UPDATE is set and a gap of more than 20 bus clock cycles is inserted between the writing of the unlock sequence values.

An attempted refresh operation between the two writes of the unlock sequence and in the WCT time following a successful unlock, goes undetected. Also, see Watchdog Operation with 8-bit access for guidelines related to 8-bit accesses to the unlock register.

Note

A context switch during unlocking and refreshing may lead to a watchdog reset.

27.3.2 Watchdog configuration time (WCT)

To prevent unintended modification of the watchdog's control and configuration register bits, you are allowed to update them only within a period of 256 bus clock cycles after unlocking. This period is known as the watchdog configuration time (WCT). In addition, these register bits can be modified only once after unlocking them for editing, even after reset.

You must unlock the registers within WCT after system reset, failing which the WDOG issues a reset to the system. In other words, you must write at least the first word of the unlocking sequence within the WCT after reset. After this is done, you have a further 20 bus clock cycles, the maximum allowed gap between the words of the unlock sequence, to complete the unlocking operation. Thereafter, to make sure that you do not forget to configure the watchdog, the watchdog issues a reset if none of the WDOG control and configuration registers is updated in the WCT after unlock. After the close of this window or after the first write, these register bits are locked out from any further changes.

The watchdog timer keeps running according to its default configuration through unlocking and update operations that can extend up to a maximum total of 2xWCT + 20 bus clock cycles. Therefore, it must be ensured that the time-out value for the watchdog is always greater than 2xWCT time + 20 bus clock cycles.

Updates in the write-once registers take effect only after the WCT window closes with the following exceptions for which changes take effect immediately:

- Stop, Wait, and Debug mode enable
- IRQ_RST_EN

The operations of refreshing the watchdog goes undetected during the WCT.

27.3.3 Refreshing the watchdog

A robust refreshing mechanism has been chosen for the watchdog. A valid refresh is a write of 0xA602 followed by 0xB480 within 20 bus clock cycles to watchdog refresh register. If these two values are written more than 20 bus cycles apart or if something other than these two values is written to the register, a watchdog reset, or interrupt-then-reset if enabled, is issued to the system. A valid refresh makes the watchdog timer restart on the next bus clock. Also, an attempted unlock operation in between the two writes of the refresh sequence goes undetected. See Watchdog Operation with 8-bit access for guidelines related to 8-bit accesses to the refresh register.

27.3.4 Windowed mode of operation

In this mode of operation, a restriction is placed on the point in time within the time-out period at which the watchdog can be refreshed. The refresh is considered valid only when the watchdog timer increments beyond a certain count as specified by the watchdog window register. This is known as refreshing the watchdog within a window of the total time-out period. If a refresh is attempted before the timer reaches the window value, the watchdog generates a reset, or interrupt-then-reset if enabled. If there is no refresh at all, the watchdog times out and generates a reset or interrupt-then-reset if enabled.

27.3.5 Watchdog disabled mode of operation

When the watchdog is disabled through the WDOG_EN bit in the watchdog status and control register, the watchdog timer is reset to zero and is disabled from counting until you enable it or it is enabled again by the system reset. In this mode, the watchdog timer cannot be refreshed–there is no requirement to do so while the timer is disabled. However, the watchdog still generates a reset, or interrupt-then-reset if enabled, on a non-time-out exception. See Generated Resets and Interrupts. You need to unlock the watchdog before enabling it. A system reset brings the watchdog out of the disabled mode.

27.3.6 Low-power modes of operation

The low-power modes of operation of the watchdog are described in the following table:

Table 27-1. Low-power modes of operation

Mode	Behavior
Wait	If the WDOG is enabled (WAIT_EN = 1), it can run on bus clock or low-power oscillator clock (CLK_SRC = x) to generate interrupt (IRQ_RST_EN=1) followed by a reset on time-out. After reset the WDOG reset counter increments by one.
Stop	Where the bus clock is gated, the WDOG can run only on low-power oscillator clock (CLK_SRC=0) if it is enabled in stop (STOP_EN=1). In this case, the WDOG runs to time-out twice, and then generates a reset from its backup circuitry. Therefore, if you program the watchdog to time-out after 100 ms and then enter such a stop mode, the reset will occur after 200 ms. Also, in this case, no interrupt will be generated irrespective of the value of IRQ_RST_EN bit. After WDOG reset, the WDOG reset counter will also not increment.
Power-Down	The watchdog is

27.3.7 Debug modes of operation

You can program the watchdog to disable in debug modes through DBG_EN in the watchdog control register. This results in the watchdog timer pausing for the duration of the mode. Register read/writes are still allowed, which means that operations like refresh, unlock, and so on are allowed. Upon exit from the mode, the timer resumes its operation from the point of pausing.

The entry of the system into the debug mode does not excuse it from compulsorily configuring the watchdog in the WCT time after unlock, unless the system bus clock is gated off, in which case the internal state machine pauses too. Failing to do so still results in a reset, or interrupt-then-reset, if enabled, to the system. Also, all of the exception conditions that result in a reset to the system, as described in Generated Resets and Interrupts, are still valid in this mode. So, if an exception condition occurs and the system bus clock is on, a reset occurs, or interrupt-then-reset, if enabled.

The entry into Debug mode within WCT after reset is treated differently. The WDOG timer is kept reset to zero and there is no need to unlock and configure it within WCT. You must not try to refresh or unlock the WDOG in this state or unknown behavior may result. Upon exit from this mode, the WDOG timer restarts and the WDOG has to be unlocked and configured within WCT.

27.4 Testing the watchdog

For IEC 60730 and other safety standards, the expectation is that anything that monitors a safety function must be tested, and this test is required to be fault tolerant. To test the watchdog, its main timer and its associated compare and reset logic must be tested. To this end, two tests are implemented for the watchdog, as described in Quick Test and

Byte Test. A control bit is provided to put the watchdog into functional test mode. There is also an overriding test-disable control bit which allows the functional test mode to be disabled permanently. After it is set, this test-disable bit can only be cleared by a reset.

These two tests achieve the overall aim of testing the counter functioning and the compare and reset logic.

Note

Do not enable the watchdog interrupt during these tests. If required, you must ensure that the effective time-out value is greater than WCT time. See Generated Resets and Interrupts for more details.

To run a particular test:

1. Select either quick test or byte test..
2. Set a certain test mode bit to put the watchdog in the functional test mode. Setting this bit automatically switches the watchdog timer to a fast clock source. The switching of the clock source is done to achieve a faster time-out and hence a faster test.

In a successful test, the timer times out after reaching the programmed time-out value and generates a system reset.

Note

After emerging from a reset due to a watchdog test, unlock and configure the watchdog. The refresh and unlock operations and interrupt are not automatically disabled in the test mode.

27.4.1 Quick test

In this test, the time-out value of watchdog timer is programmed to a very low value to achieve quick time-out. The only difference between the quick test and the normal mode of the watchdog is that TESTWDOG is set for the quick test. This allows for a faster test of the watchdog reset mechanism.

27.4.2 Byte test

The byte test is a more thorough a test of the watchdog timer. In this test, the timer is split up into its constituent byte-wide stages that are run independently and tested for time-out against the corresponding byte of the time-out value register. The following figure explains the splitting concept:

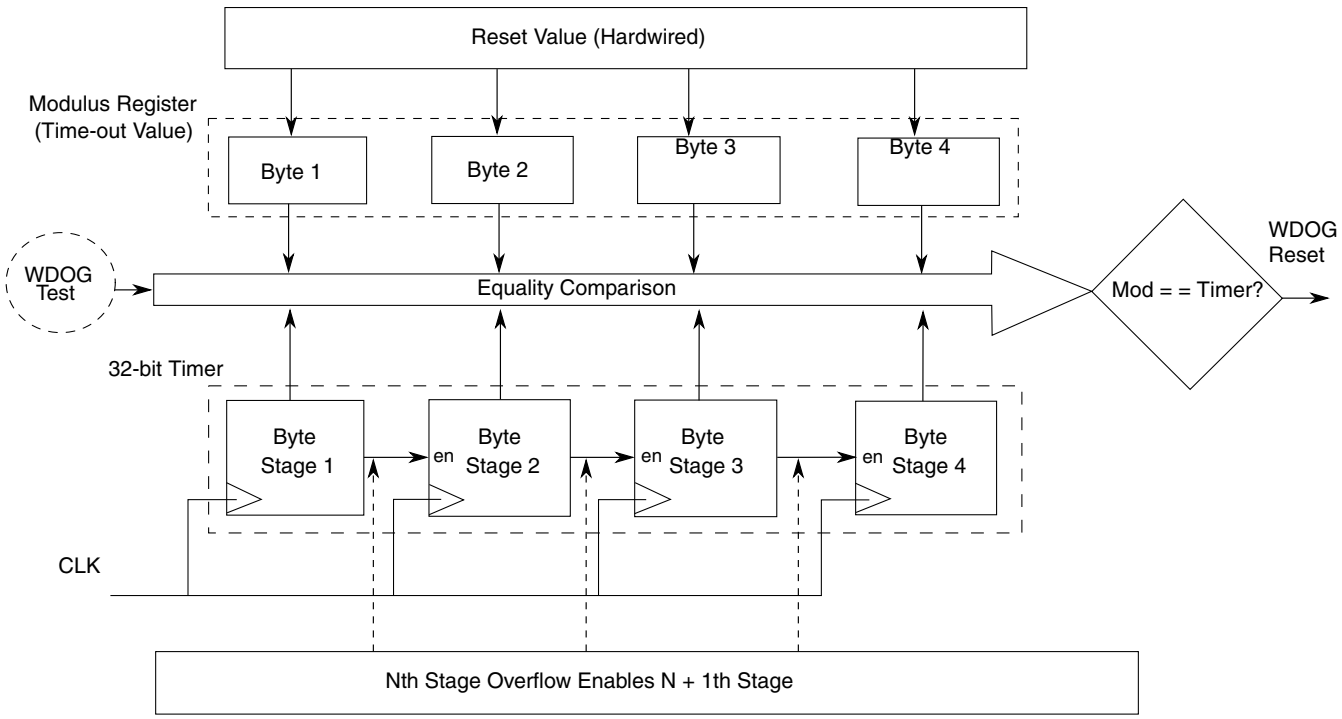


Figure 27-2. Watchdog timer byte splitting

Each stage is an 8-bit synchronous counter followed by combinational logic that generates an overflow signal. The overflow signal acts as an enable to the N + 1th stage.

In the test mode, when an individual byte, N, is tested, byte N – 1 is loaded forcefully with 0xFF, and both these bytes are allowed to run off the clock source. By doing so, the overflow signal from stage N – 1 is generated immediately, enabling counter stage N. The Nth stage runs and compares with the Nth byte of the time-out value register. In this way, the byte N is also tested along with the link between it and the preceding stage. No other stages, N – 2, N – 3... and N + 1, N + 2... are enabled for the test on byte N. These disabled stages, except the most significant stage of the counter, are loaded with a value of 0xFF.

27.5 Backup reset generator

The backup reset generator generates the final reset which goes out to the system. It has a backup mechanism which ensures that in case the bus clock stops and prevents the main state machine from generating a reset exception/interrupt, the watchdog timer's time-out is separately routed out as a reset to the system. Two successive timer time-outs without an intervening system reset result in the backup reset generator routing out the time-out signal as a reset to the system.

27.6 Generated resets and interrupts

The watchdog generates a reset in the following events, also referred to as exceptions:

- A watchdog time-out
- Failure to unlock the watchdog within WCT time after system reset deassertion
- No update of the control and configuration registers within the WCT window after unlocking. At least one of the following registers must be written to within the WCT window to avoid reset:
  - WDOG_ST_CTRL_H, WDOG_ST_CTRL_L
  - WDOG_TO_VAL_H, WDOG_TO_VAL_L
  - WDOG_WIN_H, WDOG_WIN_L
  - WDOG_PRESCALER
- A value other than the unlock sequence or the refresh sequence is written to the unlock and/or refresh registers, respectively.
- A gap of more than 20 bus cycles exists between the writes of two values of the unlock sequence.
- A gap of more than 20 bus cycles exists between the writes of two values of the refresh sequence.

The watchdog can also generate an interrupt. If IRQ_RST_EN is set, then on the above mentioned events WDOG_ST_CTRL_L[INT_FLG] is set, generating an interrupt. A watchdog reset is also generated WCT time later to ensure the watchdog is fault tolerant. The interrupt can be cleared by writing 1 to INT_FLG.

The gap of WCT between interrupt and reset means that the WDOG time-out value must be greater than WCT. Otherwise, if the interrupt was generated due to a time-out, a second consecutive time-out will occur in that WCT gap. This will trigger the backup reset generator to generate a reset to the system, prematurely ending the interrupt service routine execution. Also, jobs such as counting the number of watchdog resets would not be done.

27.7 Memory map and register definition

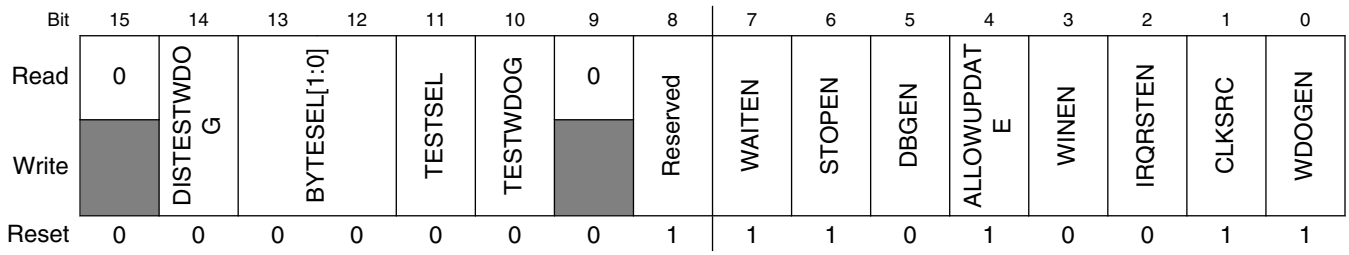
This section consists of the memory map and register descriptions.

WDOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_2000	Watchdog Status and Control Register High (WDOG_STCTRLH)	16	R/W	01D3h	27.7.1/536
4005_2002	Watchdog Status and Control Register Low (WDOG_STCTRLL)	16	R/W	0001h	27.7.2/537
4005_2004	Watchdog Time-out Value Register High (WDOG_TOVALH)	16	R/W	004Ch	27.7.3/538
4005_2006	Watchdog Time-out Value Register Low (WDOG_TOVALL)	16	R/W	4B4Ch	27.7.4/538
4005_2008	Watchdog Window Register High (WDOG_WINH)	16	R/W	0000h	27.7.5/539
4005_200A	Watchdog Window Register Low (WDOG_WINL)	16	R/W	0010h	27.7.6/539
4005_200C	Watchdog Refresh register (WDOG_REFRESH)	16	R/W	B480h	27.7.7/540
4005_200E	Watchdog Unlock register (WDOG_UNLOCK)	16	R/W	D928h	27.7.8/540
4005_2010	Watchdog Timer Output Register High (WDOG_TMROUTH)	16	R/W	0000h	27.7.9/540
4005_2012	Watchdog Timer Output Register Low (WDOG_TMROUTL)	16	R/W	0000h	27.7.10/ 541
4005_2014	Watchdog Reset Count register (WDOG_RSTCNT)	16	R/W	0000h	27.7.11/ 541
4005_2016	Watchdog Prescaler register (WDOG_PRESC)	16	R/W	0400h	27.7.12/ 541

27.7.1 Watchdog Status and Control Register High (WDOG_STCTRLH)

Address: 4005_2000h base + 0h offset = 4005_2000h



WDOG_STCTRLH field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 DISTESTWDOG	Allows the WDOG's functional test mode to be disabled permanently. After it is set, it can only be cleared by a reset. It cannot be unlocked for editing after it is set. 0 WDOG functional test mode is not disabled. 1 WDOG functional test mode is disabled permanently until reset.
13–12 BYTESEL[1:0]	This 2-bit field selects the byte to be tested when the watchdog is in the byte test mode. 00 Byte 0 selected 01 Byte 1 selected 10 Byte 2 selected 11 Byte 3 selected
11 TESTSEL	Effective only if TESTWDOG is set. Selects the test to be run on the watchdog timer. 0 Quick test. The timer runs in normal operation. You can load a small time-out value to do a quick test. 1 Byte test. Puts the timer in the byte test mode where individual bytes of the timer are enabled for operation and are compared for time-out against the corresponding byte of the programmed time-out value. Select the byte through BYTESEL[1:0] for testing.
10 TESTWDOG	Puts the watchdog in the functional test mode. In this mode, the watchdog timer and the associated compare and reset generation logic is tested for correct operation. The clock for the timer is switched from the main watchdog clock to the fast clock input for watchdog functional test. The TESTSEL bit selects the test to be run.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved.
7 WAITEN	Enables or disables WDOG in Wait mode. 0 WDOG is disabled in CPU Wait mode. 1 WDOG is enabled in CPU Wait mode.
6 STOPEN	Enables or disables WDOG in Stop mode.

Table continues on the next page...

WDOG_STCTRLH field descriptions (continued)

Field	Description
	0 WDOG is disabled in CPU Stop mode. 1 WDOG is enabled in CPU Stop mode.
5 DBGEN	Enables or disables WDOG in Debug mode. 0 WDOG is disabled in CPU Debug mode. 1 WDOG is enabled in CPU Debug mode.
4 ALLOWUPDATE	Enables updates to watchdog write-once registers, after the reset-triggered initial configuration window (WCT) closes, through unlock sequence. 0 No further updates allowed to WDOG write-once registers. 1 WDOG write-once registers can be unlocked for updating.
3 WINEN	Enables Windowing mode. 0 Windowing mode is disabled. 1 Windowing mode is enabled.
2 IRQRSTEN	Used to enable the debug breadcrumbs feature. A change in this bit is updated immediately, as opposed to updating after WCT. 0 WDOG time-out generates reset only. 1 WDOG time-out initially generates an interrupt. After WCT, it generates a reset.
1 CLKSRC	Selects clock source for the WDOG timer and other internal timing operations. 0 WDOG clock sourced from LPO . 1 WDOG clock sourced from alternate clock source.
0 WDOGEN	Enables or disables the WDOG's operation. In the disabled state, the watchdog timer is kept in the reset state, but the other exception conditions can still trigger a reset/interrupt. A change in the value of this bit must be held for more than one WDOG_CLK cycle for the WDOG to be enabled or disabled. 0 WDOG is disabled. 1 WDOG is enabled.

27.7.2 Watchdog Status and Control Register Low (WDOG_STCTRLL)

Address: 4005_2000h base + 2h offset = 4005_2002h

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

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

WDOG_STCTRLL field descriptions

Field	Description
15 INTFLG	Interrupt flag. It is set when an exception occurs. IRQRSTEN = 1 is a precondition to set this flag. INTFLG = 1 results in an interrupt being issued followed by a reset, WCT later. The interrupt can be cleared by writing 1 to this bit. It also gets cleared on a system reset.
Reserved	This field is reserved. **NOTE:** Do not modify this field value.

27.7.3 Watchdog Time-out Value Register High (WDOG_TOVALH)

Address: 4005_2000h base + 4h offset = 4005_2004h

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

WDOG_TOVALH field descriptions

Field	Description
TOVALHIGH	Defines the upper 16 bits of the 32-bit time-out value for the watchdog timer. It is defined in terms of cycles of the watchdog clock.

27.7.4 Watchdog Time-out Value Register Low (WDOG_TOVALL)

The time-out value of the watchdog must be set to a minimum of four watchdog clock cycles. This is to take into account the delay in new settings taking effect in the watchdog clock domain.

Address: 4005_2000h base + 6h offset = 4005_2006h

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

WDOG_TOVALL field descriptions

Field	Description
TOVALLOW	Defines the lower 16 bits of the 32-bit time-out value for the watchdog timer. It is defined in terms of cycles of the watchdog clock.

27.7.5 Watchdog Window Register High (WDOG_WINH)

NOTE

You must set the Window Register value lower than the Time-out Value Register.

Address: 4005_2000h base + 8h offset = 4005_2008h

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

WDOG_WINH field descriptions

Field	Description
WINHIGH	Defines the upper 16 bits of the 32-bit window for the windowed mode of operation of the watchdog. It is defined in terms of cycles of the watchdog clock. In this mode, the watchdog can be refreshed only when the timer has reached a value greater than or equal to this window length. A refresh outside this window resets the system or if IRQRSTEN is set, it interrupts and then resets the system.

27.7.6 Watchdog Window Register Low (WDOG_WINL)

NOTE

You must set the Window Register value lower than the Time-out Value Register.

Address: 4005_2000h base + Ah offset = 4005_200Ah

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

WDOG_WINL field descriptions

Field	Description
WINLOW	Defines the lower 16 bits of the 32-bit window for the windowed mode of operation of the watchdog. It is defined in terms of cycles of the pre-scaled watchdog clock. In this mode, the watchdog can be refreshed only when the timer reaches a value greater than or equal to this window length value. A refresh outside of this window resets the system or if IRQRSTEN is set, it interrupts and then resets the system.

27.7.7 Watchdog Refresh register (WDOG_REFRESH)

Address: 4005_2000h base + Ch offset = 4005_200Ch

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

WDOG_REFRESH field descriptions

Field	Description
WDOGREFRESH	Watchdog refresh register. A sequence of 0xA602 followed by 0xB480 within 20 bus clock cycles written to this register refreshes the WDOG and prevents it from resetting the system. Writing a value other than the above mentioned sequence or if the sequence is longer than 20 bus cycles, resets the system, or if IRQRSTEN is set, it interrupts and then resets the system.

27.7.8 Watchdog Unlock register (WDOG_UNLOCK)

Address: 4005_2000h base + Eh offset = 4005_200Eh

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

WDOG_UNLOCK field descriptions

Field	Description
WDOGUNLOCK	Writing the unlock sequence values to this register to makes the watchdog write-once registers writable again. The required unlock sequence is 0xC520 followed by 0xD928 within 20 bus clock cycles. A valid unlock sequence opens a window equal in length to the WCT within which you can update the registers. Writing a value other than the above mentioned sequence or if the sequence is longer than 20 bus cycles, resets the system or if IRQRSTEN is set, it interrupts and then resets the system. The unlock sequence is effective only if ALLOWUPDATE is set.

27.7.9 Watchdog Timer Output Register High (WDOG_TMROUTH)

Address: 4005_2000h base + 10h offset = 4005_2010h

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

WDOG_TMROUTH field descriptions

Field	Description
TIMEROUTHIGH	Shows the value of the upper 16 bits of the watchdog timer.

27.7.10 Watchdog Timer Output Register Low (WDOG_TMROUTL)

During Stop mode, the WDOG_TIMER_OUT will be caught at the pre-stop value of the watchdog timer. After exiting Stop mode, a maximum delay of 1 WDOG_CLK cycle + 3 bus clock cycles will occur before the WDOG_TIMER_OUT starts following the watchdog timer.

Address: 4005_2000h base + 12h offset = 4005_2012h

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

WDOG_TMROUTL field descriptions

Field	Description
TIMEROUTLOW	Shows the value of the lower 16 bits of the watchdog timer.

27.7.11 Watchdog Reset Count register (WDOG_RSTCNT)

Address: 4005_2000h base + 14h offset = 4005_2014h

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

WDOG_RSTCNT field descriptions

Field	Description
RSTCNT	Counts the number of times the watchdog resets the system. This register is reset only on a POR. Writing 1 to the bit to be cleared enables you to clear the contents of this register.

27.7.12 Watchdog Prescaler register (WDOG_PRESC)

Address: 4005_2000h base + 16h offset = 4005_2016h

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

WDOG_PRESC 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 PRESCVAL	3-bit prescaler for the watchdog clock source. A value of zero indicates no division of the input WDOG clock. The watchdog clock is divided by (PRESCVAL + 1) to provide the prescaled WDOG_CLK.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.8 Watchdog operation with 8-bit access

27.8.1 General guideline

When performing 8-bit accesses to the watchdog's 16-bit registers where the intention is to access both the bytes of a register, place the two 8-bit accesses one after the other in your code.

27.8.2 Refresh and unlock operations with 8-bit access

One exception condition that generates a reset to the system is the write of any value other than those required for a legal refresh/update sequence to the respective refresh and unlock registers.

For an 8-bit access to these registers, writing a correct value requires at least two bus clock cycles, resulting in an invalid value in the registers for one cycle. Therefore, the system is reset even if the intention is to write a correct value to the refresh/unlock register. Keeping this in mind, the exception condition for 8-bit accesses is slightly modified.

Whereas the match for a correct value for a refresh/unlock sequence is as according to the original definition, the match for an incorrect value is done byte-wise on the refresh/unlock rather than for the whole 16-bit value. This means that if the high byte of the refresh/unlock register contains any value other than high bytes of the two values that make up the sequence, it is treated as an exception condition, leading to a reset or interrupt-then-reset. The same holds true for the lower byte of the refresh or unlock register. Take the refresh operation that expects a write of 0xA602 followed by 0xB480 to the refresh register, as an example.

Table 27-2. Refresh for 8-bit access

	WDOG_REFRESH[15:8]	WDOG_REFRESH[7:0]	Sequence value1 or value2 match	Mismatch exception
Current Value	0xB4	0x80	Value2 match	No
Write 1	0xB4	0x02	No match	No
Write 2	0xA6	0x02	Value1 match	No
Write 3	0xB4	0x02	No match	No
Write 4	0xB4	0x80	Value2 match. Sequence complete.	No
Write 5	0x02	0x80	No match	Yes

As shown in the preceding table, the refresh register holds its reset value initially. Thereafter, two 8-bit accesses are performed on the register to write the first value of the refresh sequence. No mismatch exception is registered on the intermediate write, Write1. The sequence is completed by performing two more 8-bit accesses, writing in the second value of the sequence for a successful refresh. It must be noted that the match of value2 takes place only when the complete 16-bit value is correctly written, write4. Hence, the requirement of writing value2 of the sequence within 20 bus clock cycles of value1 is checked by measuring the gap between write2 and write4.

It is reiterated that the condition for matching values 1 and 2 of the refresh or unlock sequence remains unchanged. The difference for 8-bit accesses is that the criterion for detecting a mismatch is less strict. Any 16-bit access still needs to adhere to the original guidelines, mentioned in the sections Refreshing the Watchdog.

27.9 Restrictions on watchdog operation

This section mentions some exceptions to the watchdog operation that may not be apparent to you.

- Restriction on unlock/refresh operations—In the period between the closure of the WCT window after unlock and the actual reload of the watchdog timer, unlock and refresh operations need not be attempted.
- The update and reload of the watchdog timer happens two to three watchdog clocks after WCT window closes, following a successful configuration on unlock.
- Clock Switching Delay—The watchdog uses glitch-free multiplexers at two places – one to choose between the LPO oscillator input and alternate clock input, and the other to choose between the watchdog functional clock and fast clock input for

watchdog functional test. A maximum time period of ~2 clock A cycles plus ~2 clock B cycles elapses from the time a switch is requested to the occurrence of the actual clock switch, where clock A and B are the two input clocks to the clock mux.

- For the windowed mode, there is a two to three bus clock latency between the watchdog counter going past the window value and the same registering in the bus clock domain.
- For proper operation of the watchdog, the watchdog clock must be at least five times slower than the system bus clock at all times. An exception is when the watchdog clock is synchronous to the bus clock wherein the watchdog clock can be as fast as the bus clock.
- WCT must be equivalent to at least three watchdog clock cycles. If not ensured, this means that even after the close of the WCT window, you have to wait for the synchronized system reset to deassert in the watchdog clock domain, before expecting the configuration updates to take effect.
- The time-out value of the watchdog should be set to a minimum of four watchdog clock cycles. This is to take into account the delay in new settings taking effect in the watchdog clock domain.
- You must take care not only to refresh the watchdog within the watchdog timer's actual time-out period, but also provide enough allowance for the time it takes for the refresh sequence to be detected by the watchdog timer, on the watchdog clock.
- Updates cannot be made in the bus clock cycle immediately following the write of the unlock sequence, but one bus clock cycle later.
- It should be ensured that the time-out value for the watchdog is always greater than 2xWCT time + 20 bus clock cycles.
- An attempted refresh operation, in between the two writes of the unlock sequence and in the WCT time following a successful unlock, will go undetected.
- Trying to unlock the watchdog within the WCT time after an initial unlock has no effect.
- The refresh and unlock operations and interrupt are not automatically disabled in the watchdog functional test mode.
- After emerging from a reset due to a watchdog functional test, you are still expected to go through the mandatory steps of unlocking and configuring the watchdog. The watchdog continues to be in its functional test mode and therefore you should pull the watchdog out of the functional test mode within WCT time of reset.

- After emerging from a reset due to a watchdog functional test, you still need to go through the mandatory steps of unlocking and configuring the watchdog.
- You must ensure that both the clock inputs to the glitchless clock multiplexers are alive during the switching of clocks. Failure to do so results in a loss of clock at their outputs.
- There is a gap of two to three watchdog clock cycles from the point that stop mode is entered to the watchdog timer actually pausing, due to synchronization. The same holds true for an exit from the stop mode, this time resulting in a two to three watchdog clock cycle delay in the timer restarting. In case the duration of the stop mode is less than one watchdog clock cycle, the watchdog timer is not guaranteed to pause.
- Consider the case when the first refresh value is written, following which the system enters stop mode with system bus clk still on. If the second refresh value is not written within 20 bus cycles of the first value, the system is reset, or interrupt-then-reset if enabled.

Chapter 28
Multipurpose Clock Generator (MCG)

28.1 Introduction

NOTE

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

The multipurpose clock generator (MCG) module provides several clock source choices for the MCU.

The module contains a frequency-locked loop (FLL). The FLL is controllable by either an internal or an external reference clock. The module can select either an FLL output clock, or a reference clock (internal or external) as a source for the MCU system clock. The MCG operates in conjuction with a crystal oscillator, which allows an external crystal, ceramic resonator, or another external clock source to produce the external reference clock.

28.1.1 Features

Key features of the MCG module are:

- Frequency-locked loop (FLL):
  - Digitally-controlled oscillator (DCO)
  - DCO frequency range is programmable for up to four different frequency ranges.
  - Option to program and maximize DCO output frequency for a low frequency external reference clock source.
  - Option to prevent FLL from resetting its current locked frequency when switching clock modes if FLL reference frequency is not changed.

    - Internal or external reference clock can be used as the FLL source.
    - Can be used as a clock source for other on-chip peripherals.
- Internal reference clock generator:
    - Slow clock with nine trim bits for accuracy
    - Fast clock with four trim bits
    - Can be used as source clock for the FLL. In FEI mode, only the slow Internal Reference Clock (IRC) can be used as the FLL source.
    - Either the slow or the fast clock can be selected as the clock source for the MCU.
    - Can be used as a clock source for other on-chip peripherals.
- Control signals for the MCG external reference low power oscillator clock generators are provided:
    - HGO, RANGE, EREFS
- External clock from the Crystal Oscillator :
    - Can be used as a source for the FLL.
    - Can be selected as the clock source for the MCU.
- External clock monitor with reset and interrupt request capability to check for external clock failure when running in FBE, BLPE, or FEE modes
- Internal Reference Clocks Auto Trim Machine (ATM) capability using an external clock as a reference
- Reference dividers for the FLL are provided
- Reference dividers for the Fast Internal Reference Clock are provided
- MCG FLL Clock (MCGFLLCLK) is provided as a clock source for other on-chip peripherals
- MCG Fixed Frequency Clock (MCGFFCLK) is provided as a clock source for other on-chip peripherals
- MCG Internal Reference Clock (MCGIRCLK) is provided as a clock source for other on-chip peripherals

This figure presents the block diagram of the MCG module.

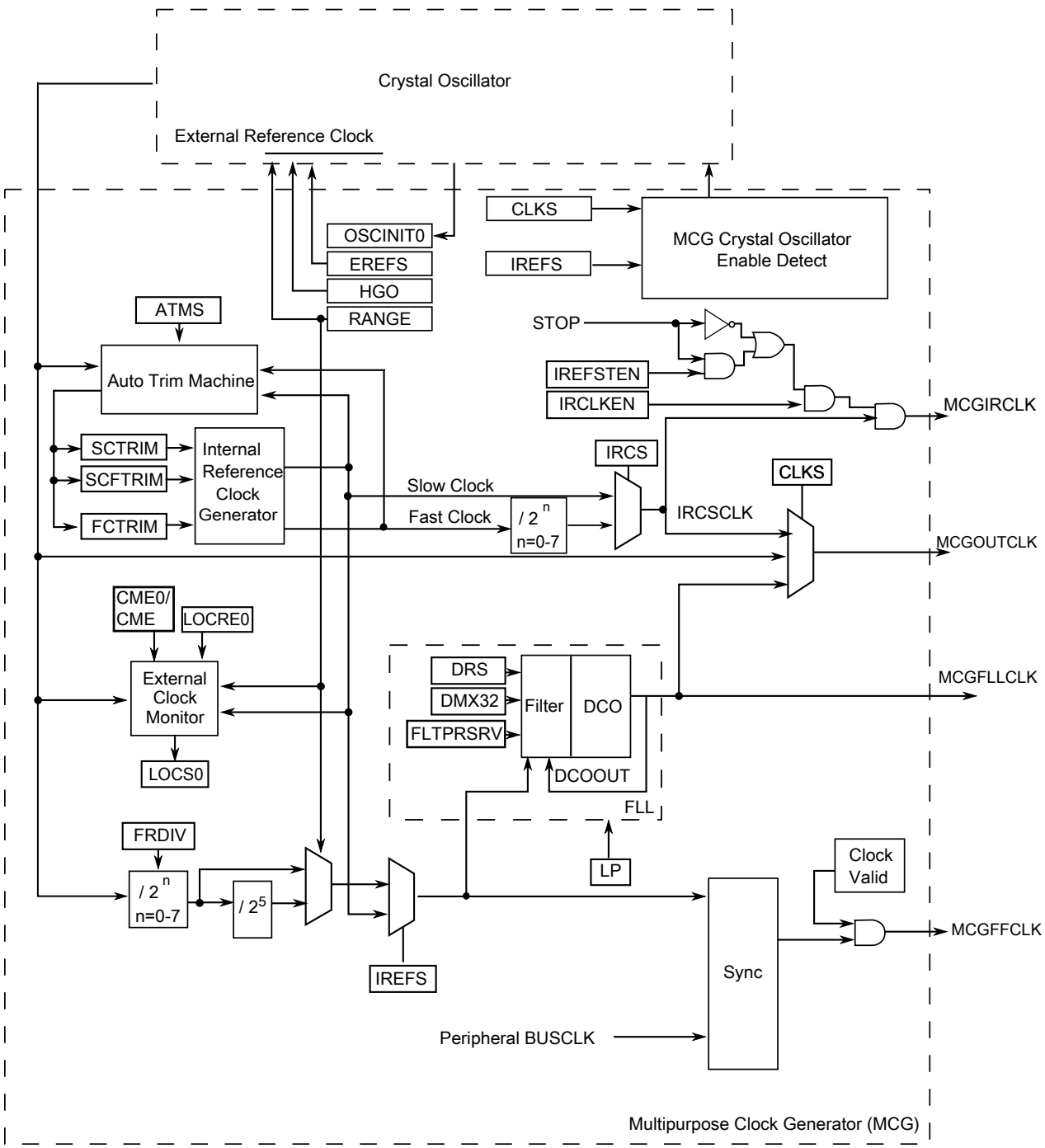


Figure 28-1. Multipurpose Clock Generator (MCG) block diagram

NOTE

Refer to the chip configuration chapter to identify the oscillator used in this MCU.

28.1.2 Modes of Operation

The MCG has the following modes of operation: FEI, FEE, FBI, FBE, BLPI, BLPE, and Stop. For details, see MCG modes of operation.

28.2 External Signal Description

There are no MCG signals that connect off chip.

28.3 Memory Map/Register Definition

This section includes the memory map and register definition.

The MCG registers can only be written when in supervisor mode. Write accesses when in user mode will result in a bus error. Read accesses may be performed in both supervisor and user mode.

MCG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_4000	MCG Control 1 Register (MCG_C1)	8	R/W	04h	28.3.1/550
4006_4001	MCG Control 2 Register (MCG_C2)	8	R/W	80h	28.3.2/551
4006_4002	MCG Control 3 Register (MCG_C3)	8	R/W	Undefined	28.3.3/553
4006_4003	MCG Control 4 Register (MCG_C4)	8	R/W	See section	28.3.4/553
4006_4004	MCG Control 5 Register (MCG_C5)	8	R/W	00h	28.3.5/554
4006_4005	MCG Control 6 Register (MCG_C6)	8	R/W	00h	28.3.5/555
4006_4006	MCG Status Register (MCG_S)	8	R	10h	28.3.6/555
4006_4008	MCG Status and Control Register (MCG_SC)	8	R/W	02h	28.3.7/556
4006_400A	MCG Auto Trim Compare Value High Register (MCG_ATCVH)	8	R/W	00h	28.3.8/558
4006_400B	MCG Auto Trim Compare Value Low Register (MCG_ATCVL)	8	R/W	00h	28.3.9/558

28.3.1 MCG Control 1 Register (MCG_C1)

Address: 4006_4000h base + 0h offset = 4006_4000h

Bit	7	6	5	4	3	2	1	0
Read Write	CLKS		FRDIV			IREFS	IRCLKEN	IREFSTEN
Reset	0	0	0	0	0	1	0	0

MCG_C1 field descriptions

Field	Description
7–6 CLKS	Clock Source Select Selects the clock source for MCGOUTCLK . 00 Encoding 0 — Output of FLL is selected. 01 Encoding 1 — Internal reference clock is selected. 10 Encoding 2 — External reference clock is selected. 11 Encoding 3 — Reserved.
5–3 FRDIV	FLL External Reference Divider Selects the amount to divide down the external reference clock for the FLL. The resulting frequency must be in the range 31.25 kHz to 39.0625 kHz (This is required when FLL/DCO is the clock source for MCGOUTCLK . In FBE mode, it is not required to meet this range, but it is recommended in the cases when trying to enter a FLL mode from FBE). 000 If RANGE = 0 , Divide Factor is 1; for all other RANGE values, Divide Factor is 32. 001 If RANGE = 0 , Divide Factor is 2; for all other RANGE values, Divide Factor is 64. 010 If RANGE = 0 , Divide Factor is 4; for all other RANGE values, Divide Factor is 128. 011 If RANGE = 0 , Divide Factor is 8; for all other RANGE values, Divide Factor is 256. 100 If RANGE = 0 , Divide Factor is 16; for all other RANGE values, Divide Factor is 512. 101 If RANGE = 0 , Divide Factor is 32; for all other RANGE values, Divide Factor is 1024. 110 If RANGE = 0 , Divide Factor is 64; for all other RANGE values, Divide Factor is 1280 . 111 If RANGE = 0 , Divide Factor is 128; for all other RANGE values, Divide Factor is 1536 .
2 IREFS	Internal Reference Select Selects the reference clock source for the FLL. 0 External reference clock is selected. 1 The slow internal reference clock is selected.
1 IRCLKEN	Internal Reference Clock Enable Enables the internal reference clock for use as MCGIRCLK. 0 MCGIRCLK inactive. 1 MCGIRCLK active.
0 IREFSTEN	Internal Reference Stop Enable Controls whether or not the internal reference clock remains enabled when the MCG enters Stop mode. 0 Internal reference clock is disabled in Stop mode. 1 Internal reference clock is enabled in Stop mode if IRCLKEN is set or if MCG is in FEI, FBI, or BLPI modes before entering Stop mode.

28.3.2 MCG Control 2 Register (MCG_C2)

Address: 4006_4000h base + 1h offset = 4006_4001h

Bit	7	6	5	4	3	2	1	0
Read Write	LOCRE0	FCFTRIM	RANGE		HGO	EREFS	LP	IRCS
Reset	1	0	0	0	0	0	0	0

MCG_C2 field descriptions

Field	Description
7 LOCRE0	Loss of Clock Reset Enable Determines whether an interrupt or a reset request is made following a loss of OSC0 external reference clock. The LOCRE0 only has an affect when CME0 is set. 0 Interrupt request is generated on a loss of OSC0 external reference clock. 1 Generate a reset request on a loss of OSC0 external reference clock.
6 FCFTRIM	Fast Internal Reference Clock Fine Trim FCFTRIM controls the smallest adjustment of the fast internal reference clock frequency. Setting FCFTRIM increases the period and clearing FCFTRIM decreases the period by the smallest amount possible. If an FCFTRIM value stored in nonvolatile memory is to be used, it is your responsibility to copy that value from the nonvolatile memory location to this bit.
5–4 RANGE	Frequency Range Select Selects the frequency range for the crystal oscillator or external clock source. See the Oscillator (OSC) chapter for more details and the device data sheet for the frequency ranges used. 00 Encoding 0 — Low frequency range selected for the crystal oscillator . 01 Encoding 1 — High frequency range selected for the crystal oscillator . 1X Encoding 2 — Very high frequency range selected for the crystal oscillator .
3 HGO	High Gain Oscillator Select Controls the crystal oscillator mode of operation. See the Oscillator (OSC) chapter for more details. 0 Configure crystal oscillator for low-power operation. 1 Configure crystal oscillator for high-gain operation.
2 EREFS	External Reference Select Selects the source for the external reference clock. See the Oscillator (OSC) chapter for more details. 0 External reference clock requested. 1 Oscillator requested.
1 LP	Low Power Select Controls whether the FLL is disabled in BLPI and BLPE modes. In FBE mode, setting this bit to 1 will transition the MCG into BLPE mode; in FBI mode, setting this bit to 1 will transition the MCG into BLPI mode. In any other MCG mode, LP bit has no affect. 0 FLL is not disabled in bypass modes. 1 FLL is disabled in bypass modes (lower power)
0 IRCS	Internal Reference Clock Select Selects between the fast or slow internal reference clock source. 0 Slow internal reference clock selected. 1 Fast internal reference clock selected.

28.3.3 MCG Control 3 Register (MCG_C3)

Address: 4006_4000h base + 2h offset = 4006_4002h

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

* Notes:
- x = Undefined at reset.

MCG_C3 field descriptions

Field	Description
SCTRIM	Slow Internal Reference Clock Trim Setting SCTRIM [1] controls the slow internal reference clock frequency by controlling the slow internal reference clock period. The SCTRIM bits are binary weighted, that is, bit 1 adjusts twice as much as bit 0. Increasing the binary value increases the period, and decreasing the value decreases the period. An additional fine trim bit is available in C4 register as the SCFTRIM bit. Upon reset, this value is loaded with a factory trim value. If an SCTRIM value stored in nonvolatile memory is to be used, it is your responsibility to copy that value from the nonvolatile memory location to this register.

1. A value for SCTRIM is loaded during reset from a factory programmed location.

28.3.4 MCG Control 4 Register (MCG_C4)

Address: 4006_4000h base + 3h offset = 4006_4003h

Bit	7	6	5	4	3	2	1	0
Read Write	DMX32	DRST_DRS		FCTRIM				SCFTRIM
Reset	0	0	0	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

MCG_C4 field descriptions

Field	Description
7 DMX32	DCO Maximum Frequency with 32.768 kHz Reference The DMX32 bit controls whether the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference. The following table identifies settings for the DCO frequency range. **NOTE:** The system clocks derived from this source should not exceed their specified maximums.

DRST_DRS	DMX32	Reference Range	FLL Factor	DCO Range
00	0	31.25–39.0625 kHz	640	20–25 MHz

Table continues on the next page...

MCG_C4 field descriptions (continued)

Field	Description
	1 | 32.768 kHz | 732 | 24 MHz 01 | 0 | 31.25–39.0625 kHz | 1280 | 40–50 MHz 1 | 32.768 kHz | 1464 | 48 MHz 10 | 0 | 31.25–39.0625 kHz | 1920 | 60–75 MHz 1 | 32.768 kHz | 2197 | 72 MHz 11 | 0 | 31.25–39.0625 kHz | 2560 | 80–100 MHz 1 | 32.768 kHz | 2929 | 96 MHz 0 DCO has a default range of 25%. 1 DCO is fine-tuned for maximum frequency with 32.768 kHz reference.
6–5 DRST_DRS	DCO Range Select The DRS bits select the frequency range for the FLL output, DCOOUT. When the LP bit is set, writes to the DRS bits are ignored. The DRST read field indicates the current frequency range for DCOOUT. The DRST field does not update immediately after a write to the DRS field due to internal synchronization between clock domains. See the DCO Frequency Range table for more details. 00 Encoding 0 — Low range (reset default). 01 Encoding 1 — Mid range. 10 Encoding 2 — Mid-high range. 11 Encoding 3 — High range.
4–1 FCTRIM	Fast Internal Reference Clock Trim Setting FCTRIM [1] controls the fast internal reference clock frequency by controlling the fast internal reference clock period. The FCTRIM bits are binary weighted, that is, bit 1 adjusts twice as much as bit 0. Increasing the binary value increases the period, and decreasing the value decreases the period. If an FCTRIM[3:0] value stored in nonvolatile memory is to be used, it is your responsibility to copy that value from the nonvolatile memory location to this register.
0 SCFTRIM	Slow Internal Reference Clock Fine Trim SCFTRIM [2] controls the smallest adjustment of the slow internal reference clock frequency. Setting SCFTRIM increases the period and clearing SCFTRIM decreases the period by the smallest amount possible. If an SCFTRIM value stored in nonvolatile memory is to be used, it is your responsibility to copy that value from the nonvolatile memory location to this bit.

1. A value for FCTRIM is loaded during reset from a factory programmed location.
2. A value for SCFTRIM is loaded during reset from a factory programmed location.

28.3.5 MCG Control 5 Register (MCG_C5)

Address: 4006_4000h base + 4h offset = 4006_4004h

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

MCG_C5 field descriptions

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

28.3.5 MCG Control 6 Register (MCG_C6)

Address: 4006_4000h base + 5h offset = 4006_4005h

Bit	7	6	5	4	3	2	1	0
Read	0		CME0		0			
Write								
Reset	0	0	0	0	0	0	0	0

MCG_C6 field descriptions

Field	Description
7–6 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
5 CME0	Clock Monitor Enable Determines if an interrupt or a reset request (see MCG_C2[LOCRE0]) is made following a loss of external clock indication. The CME0 bit should only be set to a logic 1 when the MCG is in an operational mode that uses the external clock (FEE, FBE, or BLPE). Whenever the CME0 bit is set to a logic 1, the value of the RANGE bits in the C2 register should not be changed. CME0 bit should be set to a logic 0 before the MCG enters any Stop mode. Otherwise, a reset request may occur when in Stop mode. CME0 should also be set to a logic 0 before entering VLPR or VLPW power modes if the MCG is in BLPE mode. 0 External clock monitor is disabled. 1 Generate an interrupt or a reset request (see MCG_C2[LOCRE0]) on loss of external clock.
Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.

28.3.6 MCG Status Register (MCG_S)

Address: 4006_4000h base + 6h offset = 4006_4006h

Bit	7	6	5	4	3	2	1	0
Read	0			IREFST	CLKST		OSCINIT0	IRCST
Write								
Reset	0	0	0	1	0	0	0	0

MCG_S field descriptions

Field	Description
7–5 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
4 IREFST	Internal Reference Status This bit indicates the current source for the FLL reference clock. The IREFST bit does not update immediately after a write to the IREFS bit due to internal synchronization between clock domains. 0 Source of FLL reference clock is the external reference clock. 1 Source of FLL reference clock is the internal reference clock.
3–2 CLKST	Clock Mode Status These bits indicate the current clock mode. The CLKST bits do not update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Encoding 0 — Output of the FLL is selected (reset default). 01 Encoding 1 — Internal reference clock is selected. 10 Encoding 2 — External reference clock is selected. 11 Reserved.
1 OSCINIT0	OSC Initialization This bit, which resets to 0, is set to 1 after the initialization cycles of the crystal oscillator clock have completed. After being set, the bit is cleared to 0 if the OSC is subsequently disabled. See the OSC module's detailed description for more information.
0 IRCST	Internal Reference Clock Status The IRCST bit indicates the current source for the internal reference clock select clock (IRCSCLK). The IRCST bit does not update immediately after a write to the IRCS bit due to internal synchronization between clock domains. The IRCST bit will only be updated if the internal reference clock is enabled, either by the MCG being in a mode that uses the IRC or by setting the C1[IRCLKEN] bit . 0 Source of internal reference clock is the slow clock (32 kHz IRC). 1 Source of internal reference clock is the fast clock (4 MHz IRC).

28.3.7 MCG Status and Control Register (MCG_SC)

Address: 4006_4000h base + 8h offset = 4006_4008h

Bit	7	6	5	4	3	2	1	0
Read	ATME	ATMS	ATMF	FLTPRSRV	FCRDIV			LOCS0
Write			w1c					w1c
Reset	0	0	0	0	0	0	1	0

MCG_SC field descriptions

Field	Description
7 ATME	Automatic Trim Machine Enable

Table continues on the next page...

MCG_SC field descriptions (continued)

Field	Description
	Enables the Auto Trim Machine to start automatically trimming the selected Internal Reference Clock. **NOTE:** ATME deasserts after the Auto Trim Machine has completed trimming all trim bits of the IRCS clock selected by the ATMS bit. Writing to C1, C3, C4, and SC registers or entering Stop mode aborts the auto trim operation and clears this bit. 0 Auto Trim Machine disabled. 1 Auto Trim Machine enabled.
6 ATMS	Automatic Trim Machine Select Selects the IRCS clock for Auto Trim Test. 0 32 kHz Internal Reference Clock selected. 1 4 MHz Internal Reference Clock selected.
5 ATMF	Automatic Trim Machine Fail Flag Fail flag for the Automatic Trim Machine (ATM). This bit asserts when the Automatic Trim Machine is enabled, ATME=1, and a write to the C1, C3, C4, and SC registers is detected or the MCG enters into any Stop mode. A write to ATMF clears the flag. 0 Automatic Trim Machine completed normally. 1 Automatic Trim Machine failed.
4 FLTPRSRV	FLL Filter Preserve Enable This bit will prevent the FLL filter values from resetting allowing the FLL output frequency to remain the same during clock mode changes where the FLL/DCO output is still valid. (Note: This requires that the FLL reference frequency to remain the same as what it was prior to the new clock mode switch. Otherwise FLL filter and frequency values will change.) 0 FLL filter and FLL frequency will reset on changes to currect clock mode. 1 Fll filter and FLL frequency retain their previous values during new clock mode change.
3–1 FCRDIV	Fast Clock Internal Reference Divider Selects the amount to divide down the fast internal reference clock. The resulting frequency will be in the range 31.25 kHz to 4 MHz (Note: Changing the divider when the Fast IRC is enabled is not supported). 000 Divide Factor is 1 001 Divide Factor is 2. 010 Divide Factor is 4. 011 Divide Factor is 8. 100 Divide Factor is 16 101 Divide Factor is 32 110 Divide Factor is 64 111 Divide Factor is 128.
0 LOCS0	OSC0 Loss of Clock Status The LOCS0 indicates when a loss of OSC0 reference clock has occurred. The LOCS0 bit only has an effect when CME0 is set. This bit is cleared by writing a logic 1 to it when set. 0 Loss of OSC0 has not occurred. 1 Loss of OSC0 has occurred.

28.3.8 MCG Auto Trim Compare Value High Register (MCG_ATCVH)

Address: 4006_4000h base + Ah offset = 4006_400Ah

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

MCG_ATCVH field descriptions

Field	Description
ATCVH	ATM Compare Value High Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion.

28.3.9 MCG Auto Trim Compare Value Low Register (MCG_ATCVL)

Address: 4006_4000h base + Bh offset = 4006_400Bh

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

MCG_ATCVL field descriptions

Field	Description
ATCVL	ATM Compare Value Low Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion.

28.4 Functional description

28.4.1 MCG mode state diagram

The seven states of the MCG are shown in the following figure and are described in Table 28-1. The arrows indicate the permitted MCG mode transitions.

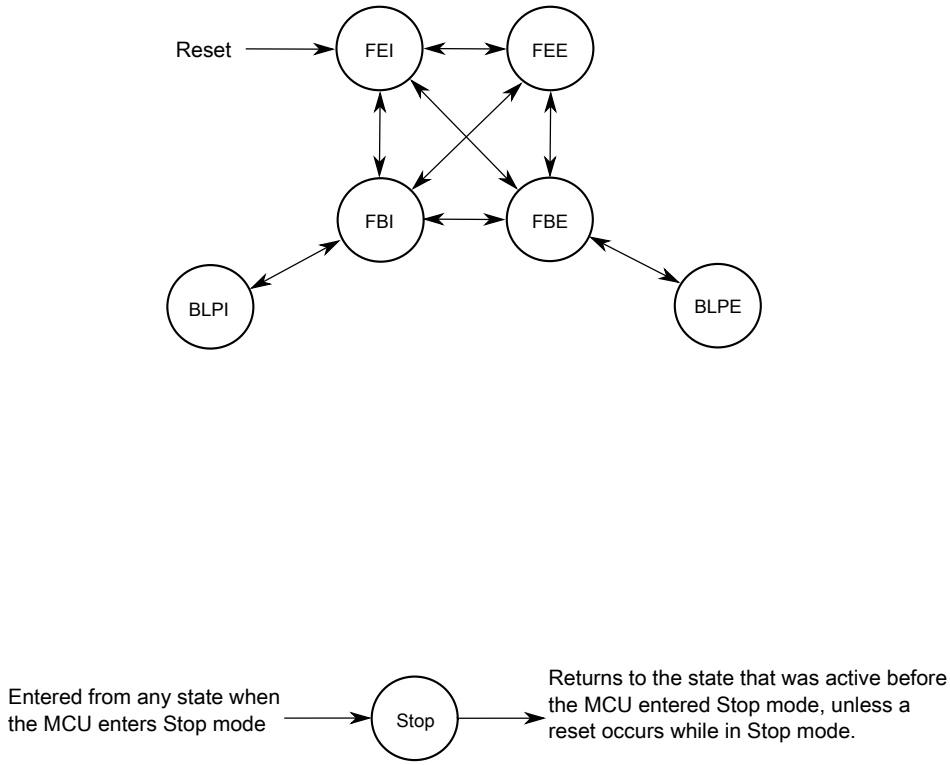


Figure 28-2. MCG mode state diagram

28.4.1.1 MCG modes of operation

The MCG operates in one of the following modes.

Note

The MCG restricts transitions between modes. For the permitted transitions, see Figure 28-2.

Table 28-1. MCG modes of operation

Mode	Description
FLL Engaged Internal (FEI)	FLL engaged internal (FEI) is the default mode of operation and is entered when all the following condtions occur: • 00 is written to C1[CLKS]. • 1 is written to C1[IREFS]. In FEI mode, MCGOUTCLK is derived from the FLL clock (DCOCLK) that is controlled by the 32 kHz Internal Reference Clock (IRC). The FLL loop will lock the DCO frequency to the FLL factor, as selected by C4[DRST_DRS] and C4[DMX32] bits, times the internal reference frequency. See the C4[DMX32] bit description for more details.
FLL Engaged External (FEE)	FLL engaged external (FEE) mode is entered when all the following conditions occur: • 00 is written to C1[CLKS].

Table continues on the next page...

Table 28-1. MCG modes of operation (continued)

Mode	Description
	• 0 is written to C1[IREFS]. • C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz In FEE mode, MCGOUTCLK is derived from the FLL clock (DCOCLK) that is controlled by the external reference clock. The FLL loop will lock the DCO frequency to the FLL factor, as selected by C4[DRST_DRS] and C4[DMX32] bits, times the external reference frequency, as specified by C1[FRDIV] and C2[RANGE]. See the C4[DMX32] bit description for more details.
FLL Bypassed Internal (FBI)	FLL bypassed internal (FBI) mode is entered when all the following conditions occur: • 01 is written to C1[CLKS]. • 1 is written to C1[IREFS]. • 0 is written to C2[LP]. In FBI mode, the MCGOUTCLK is derived either from the slow (32 kHz IRC) or fast (4 MHz IRC) internal reference clock, as selected by the C2[IRCS] bit. The FLL is operational but its output is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUTCLK is driven from the C2[IRCS] selected internal reference clock. The FLL clock (DCOCLK) is controlled by the slow internal reference clock, and the DCO clock frequency locks to a multiplication factor, as selected by C4[DRST_DRS] and C4[DMX32] bits, times the internal reference frequency. See the C4[DMX32] bit description for more details.
FLL Bypassed External (FBE)	FLL bypassed external (FBE) mode is entered when all the following conditions occur: • 10 is written to C1[CLKS]. • 0 is written to C1[IREFS]. • C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz. • 0 is written to C2[LP]. In FBE mode, the MCGOUTCLK is derived from the external reference clock. The FLL is operational but its output is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUTCLK is driven from the external reference clock. The FLL clock (DCOCLK) is controlled by the external reference clock, and the DCO clock frequency locks to a multiplication factor, as selected by C4[DRST_DRS] and C4[DMX32] bits, times the divided external reference frequency. See the C4[DMX32] bit description for more details.
Bypassed Low Power Internal (BLPI) [1]	Bypassed Low Power Internal (BLPI) mode is entered when all the following conditions occur: • 01 is written to C1[CLKS]. • 1 is written to C1[IREFS]. • 1 is written to C2[LP]. In BLPI mode, MCGOUTCLK is derived from the internal reference clock. The FLL is disabled
Bypassed Low Power External (BLPE) [1]	Bypassed Low Power External (BLPE) mode is entered when all the following conditions occur: • 10 is written to C1[CLKS]. • 0 is written to C1[IREFS]. • 1 is written to C2[LP]. In BLPE mode, MCGOUTCLK is derived from the external reference clock. The FLL is disabled

Table continues on the next page...

Table 28-1. MCG modes of operation (continued)

Mode	Description
Stop	Entered whenever the MCU enters a Stop state. The power modes are chip specific. For power mode assignments, see the Power management chapter that describes how modules are configured and MCG behavior during Stop recovery. Entering Stop mode, the FLL is disabled, and all MCG clock signals are static except in the following case: MCGIRCLK is active in Normal Stop mode when all the following conditions become true: • C1[IRCLKEN] = 1 • C1[IREFSTEN] = 1 **NOTE:** • In VLPS Stop Mode, the MCGIRCLK can be programmed to stay enabled and continue running if C1[IRCLKEN] = 1, C1[IREFSTEN]=1, and Fast IRC clock is selected (C2[IRCS] = 1)

1. **Caution:** If entering VLPR mode, MCG has to be configured and enter BLPE mode or BLPI mode with the Fast IRC clock selected (C2[IRCS]=1). After it enters VLPR mode, writes to any of the MCG control registers that can cause an MCG clock mode switch to a non low power clock mode must be avoided.

NOTE

For the chip-specific modes of operation, see the power management chapter of this MCU.

28.4.1.2 MCG mode switching

C1[IREFS] can be changed at any time, but the actual switch to the newly selected reference clocks is shown by S[IREFST]. When switching between engaged internal and engaged external modes, the FLL will begin locking again after the switch is completed.

C1[CLKS] can also be changed at any time, but the actual switch to the newly selected clock is shown by S[CLKST]. If the newly selected clock is not available, the previous clock will remain selected.

The C4[DRST_DRS] write bits can be changed at any time except when C2[LP] bit is 1. If C4[DRST_DRS] write bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE) mode, the MCGOUTCLK switches to the new selected DCO range within three clocks of the selected DCO clock. After switching to the new DCO (indicated by the updated C4[DRST_DRS] read bits), the FLL remains unlocked for several reference cycles. The FLL lock time is provided in the device data sheet as $t_{fll\_acquire}$.

28.4.2 Low-power bit usage

C2[LP] is provided to allow the FLL to be disabled and thus conserve power when these systems are not being used. C4[DRST_DRS] can not be written while C2[LP] is 1. However, in some applications, it may be desirable to enable the FLL and allow it to lock for maximum accuracy before switching to an engaged mode. Do this by writing 0 to C2[LP].

28.4.3 MCG Internal Reference Clocks

This module supports two internal reference clocks with nominal frequencies of 32 kHz (slow IRC) and 4 MHz (fast IRC). The fast IRC frequency can be divided down by programming of the FCRDIV to produce a frequency range of 32 kHz to 4 MHz.

28.4.3.1 MCG Internal Reference Clock

The MCG Internal Reference Clock (MCGIRCLK) provides a clock source for other on-chip peripherals and is enabled when C1[IRCLKEN]=1. When enabled, MCGIRCLK is driven by either the fast internal reference clock (4 MHz IRC which can be divided down by the FRDIV factors) or the slow internal reference clock (32 kHz IRC). The IRCS clock frequency can be re-targeted by trimming the period of its IRCS selected internal reference clock. This can be done by writing a new trim value to the C3[SCTRIM]:C4[SCFTRIM] bits when the slow IRC clock is selected or by writing a new trim value to C4[FCTRIM]:C2[FCFTRIM] when the fast IRC clock is selected. The internal reference clock period is proportional to the trim value written. C3[SCTRIM]:C4[SCFTRIM] (if C2[IRCS]=0) and C4[FCTRIM]:C2[FCFTRIM] (if C2[IRCS]=1) bits affect the MCGOUTCLK frequency if the MCG is in FBI or BLPI modes. C3[SCTRIM]:C4[SCFTRIM] (if C2[IRCS]=0) bits also affect the MCGOUTCLK frequency if the MCG is in FEI mode.

Additionally, this clock can be enabled in Stop mode by setting C1[IRCLKEN] and C1[IREFSTEN], otherwise this clock is disabled in Stop mode.

28.4.4 External Reference Clock

The MCG module can support an external reference clock in all modes. See the device datasheet for external reference frequency range. When C1[IREFS] is set, the external reference clock will not be used by the FLL. In these mode, the frequency can be equal to the maximum frequency the chip-level timing specifications will support.

If any of the CME bits are asserted the slow internal reference clock is enabled along with the enabled external clock monitor. For the case when C6[CME0]=1, a loss of clock is detected if the OSC0 external reference falls below a minimum frequency (f_{loc_high} or f_{loc_low} depending on C2[RANGE0]).

NOTE

All clock monitors must be disabled before entering these low-power modes: Stop, VLPS, VLPR, VLPW, and VLLSx.

On detecting a loss-of-clock event, the MCU generates a system reset if the respective LOCRE bit is set. Otherwise the MCG sets the respective LOCS bit and the MCG generates a LOCS interrupt request.

28.4.5 MCG Fixed Frequency Clock

The MCG Fixed Frequency Clock (MCGFFCLK) provides a fixed frequency clock source for other on-chip peripherals; see the block diagram. This clock is driven by either the slow clock from the internal reference clock generator or the external reference clock from the Crystal Oscillator, divided by the FLL reference clock divider. The source of MCGFFCLK is selected by C1[IREFS].

This clock is synchronized to the peripheral bus clock and is valid only when its frequency is not more than 1/8 of the MCGOUTCLK frequency. When it is not valid, it is disabled and held high. The MCGFFCLK is not available when the MCG is in BLPI mode. This clock is also disabled in Stop mode. The FLL reference clock must be set within the valid frequency range for the MCGFFCLK.

28.4.6 MCG Auto TRIM (ATM)

The MCG Auto Trim (ATM) is a MCG feature that when enabled, it configures the MCG hardware to automatically trim the MCG Internal Reference Clocks using an external clock as a reference. The selection between which MCG IRC clock gets tested and enabled is controlled by the ATC[ATMS] control bit (ATC[ATMS]=0 selects the 32 kHz IRC and ATC[ATMS]=1 selects the 4 MHz IRC). If 4 MHz IRC is selected for the ATM, a divide by 128 is enabled to divide down the 4 MHz IRC to a range of 31.250 kHz.

When MCG ATM is enabled by writing ATC[ATME] bit to 1, The ATM machine will start auto trimming the selected IRC clock. During the autotrim process, ATC[ATME] will remain asserted and will deassert after ATM is completed or an abort occurs. The

MCG ATM is aborted if a write to any of the following control registers is detected : C1, C3, C4, or ATC or if Stop mode is entered. If an abort occurs, ATC[ATMF] fail flag is asserted.

The ATM machine uses the bus clock as the external reference clock to perform the IRC auto-trim. Therefore, it is required that the MCG is configured in a clock mode where the reference clock used to generate the system clock is the external reference clock such as FBE clock mode. The MCG must not be configured in a clock mode where selected IRC ATM clock is used to generate the system clock. The bus clock is also required to be running with in the range of 8–16 MHz.

To perform the ATM on the selected IRC, the ATM machine uses the successive approximation technique to adjust the IRC trim bits to generate the desired IRC trimmed frequency. The ATM SARs each of the ATM IRC trim bits starting with the MSB. For each trim bit test, the ATM uses a pulse that is generated by the ATM selected IRC clock to enable a counter that counts number of ATM external clocks. At end of each trim bit, the ATM external counter value is compared to the ATCV[15:0] register value. Based on the comparison result, the ATM trim bit under test will get cleared or stay asserted. This is done until all trim bits have been tested by ATM SAR machine.

Before the ATM can be enabled, the ATM expected count needs to be derived and stored into the ATCV register. The ATCV expected count is derived based on the required target Internal Reference Clock (IRC) frequency, and the frequency of the external reference clock using the following formula:

$$\text{ATCV Expected Count Value} = 21 * (Fe / Fr)$$

- Fr = Target Internal Reference Clock (IRC) Trimmed Frequency
- Fe = External Clock Frequency

If the auto trim is being performed on the 4 MHz IRC, the calculated expected count value must be multiplied by 128 before storing it in the ATCV register. Therefore, the ATCV Expected Count Value for trimming the 4 MHz IRC is calculated using the following formula.

$$\text{Expected Count Value} = (Fe / Fr) * 21 * (128)$$

28.5 Initialization / Application information

This section describes how to initialize and configure the MCG module in an application.

The following sections include examples on how to initialize the MCG and properly switch between the various available modes.

28.5.1 MCG module initialization sequence

The MCG comes out of reset configured for FEI mode.

The internal reference will stabilize in $t_{irefsts}$ microseconds before the FLL can acquire lock. As soon as the internal reference is stable, the FLL will acquire lock in $t_{fll\_acquire}$ milliseconds.

28.5.1.1 Initializing the MCG

Because the MCG comes out of reset in FEI mode, the only MCG modes that can be directly switched to upon reset are FEE, FBE, and FBI modes (see Figure 28-2). Reaching any of the other modes requires first configuring the MCG for one of these three intermediate modes. Care must be taken to check relevant status bits in the MCG status register reflecting all configuration changes within each mode.

To change from FEI mode to FEE or FBE modes, follow this procedure:

1. Enable the external clock source by setting the appropriate bits in C2 register.
2. Write to C1 register to select the clock mode.
   - If entering FEE mode, set C1[FRDIV] appropriately, clear C1[IREFS] bit to switch to the external reference, and leave C1[CLKS] at 2'b00 so that the output of the FLL is selected as the system clock source.
   - If entering FBE, clear C1[IREFS] to switch to the external reference and change C1[CLKS] to 2'b10 so that the external reference clock is selected as the system clock source. The C1[FRDIV] bits should also be set appropriately here according to the external reference frequency to keep the FLL reference clock in the range of 31.25 kHz to 39.0625 kHz. Although the FLL is bypassed, it is still on in FBE mode.
   - The internal reference can optionally be kept running by setting C1[IRCLKEN]. This is useful if the application will switch back and forth between internal and external modes. For minimum power consumption, leave the internal reference disabled while in an external clock mode.
3. Once the proper configuration bits have been set, wait for the affected bits in the MCG status register to be changed appropriately, reflecting that the MCG has moved into the proper mode.

- If the MCG is in FEE, FBE, or BLPE mode, and C2[EREFS] was also set in step 1, wait here for S[OSCINIT0] bit to become set indicating that the external clock source has finished its initialization cycles and stabilized.
- If in FEE mode, check to make sure S[IREFST] is cleared before moving on.
- If in FBE mode, check to make sure S[IREFST] is cleared and S[CLKST] bits have changed to 2'b10 indicating the external reference clock has been appropriately selected. Although the FLL is bypassed, it is still on in FBE mode.

4. Write to the C4 register to determine the DCO output (MCGFLLCLK) frequency range.
   - By default, with C4[DMX32] cleared to 0, the FLL multiplier for the DCO output is 640. For greater flexibility, if a mid-low-range FLL multiplier of 1280 is desired instead, set C4[DRST_DRS] bits to 2'b01 for a DCO output frequency of 40 MHz. If a mid high-range FLL multiplier of 1920 is desired instead, set the C4[DRST_DRS] bits to 2'b10 for a DCO output frequency of 60 MHz. If a high-range FLL multiplier of 2560 is desired instead, set the C4[DRST_DRS] bits to 2'b11 for a DCO output frequency of 80 MHz.
   - When using a 32.768 kHz external reference, if the maximum low-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b00 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 732 will be 24 MHz.
   - When using a 32.768 kHz external reference, if the maximum mid-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b01 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 1464 will be 48 MHz.
   - When using a 32.768 kHz external reference, if the maximum mid high-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b10 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 2197 will be 72 MHz.
   - When using a 32.768 kHz external reference, if the maximum high-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set C4[DRST_DRS] bits to 2'b11 and set C4[DMX32] bit to 1. The resulting DCO output (MCGOUTCLK) frequency with the new multiplier of 2929 will be 96 MHz.

5. Wait for the FLL lock time to guarantee FLL is running at new C4[DRST_DRS] and C4[DMX32] programmed frequency.

To change from FEI clock mode to FBI clock mode, follow this procedure:

1. Change C1[CLKS] bits in C1 register to 2'b01 so that the internal reference clock is selected as the system clock source.
2. Wait for S[CLKST] bits in the MCG status register to change to 2'b01, indicating that the internal reference clock has been appropriately selected.
3. Write to the C2 register to determine the IRCS output (IRCSCLK) frequency range.
    - By default, with C2[IRCS] cleared to 0, the IRCS selected output clock is the slow internal reference clock (32 kHz IRC). If the faster IRC is desired, set C2[IRCS] to 1 for a IRCS clock derived from the 4 MHz IRC source.

28.5.2 Using a 32.768 kHz reference

In FEE and FBE modes, if using a 32.768 kHz external reference, at the default FLL multiplication factor of 640, the DCO output (MCGFLLCLK) frequency is 20.97 MHz at low-range.

If C4[DRST_DRS] bits are set to 2'b01, the multiplication factor is doubled to 1280, and the resulting DCO output frequency is 41.94 MHz at mid-low-range. If C4[DRST_DRS] bits are set to 2'b10, the multiplication factor is set to 1920, and the resulting DCO output frequency is 62.91 MHz at mid high-range. If C4[DRST_DRS] bits are set to 2'b11, the multiplication factor is set to 2560, and the resulting DCO output frequency is 83.89 MHz at high-range.

In FBI and FEI modes, setting C4[DMX32] bit is not recommended. If the internal reference is trimmed to a frequency above 32.768 kHz, the greater FLL multiplication factor could potentially push the microcontroller system clock out of specification and damage the part.

28.5.3 MCG mode switching

When switching between operational modes of the MCG, certain configuration bits must be changed in order to properly move from one mode to another.

Each time any of these bits are changed (C1[IREFS], C1[CLKS], C2[IRCS], or C2[EREFS], the corresponding bits in the MCG status register (IREFST, CLKST, IRCST, or OSCINIT) must be checked before moving on in the application software.

Additionally, care must be taken to ensure that the reference clock divider (C1[FRDIV]) is set properly for the mode being switched to. For instance, in FEE mode, if using a 4MHz crystal, C1[FRDIV] must be set to 3'b010 (divide-by-128) to devide the external frequency down to the required frequency between 31.25 and 39.0625 kHz.

In FBE, FEE, FBI, and FEI modes, at any time, the application can switch the FLL multiplication factor between 640, 1280, 1920, and 2560 with C4[DRST_DRS] bits. Writes to C4[DRST_DRS] bits will be ignored if C2[LP]=1.

The table below shows MCGOUTCLK frequency calculations using C1[FRDIV]settings for each clock mode.

Table 28-2. MCGOUTCLK Frequency Calculation Options

Clock Mode	$f_{MCGOUTCLK}$[1]	Note
FEI (FLL engaged internal)	$f_{int} \times F$	Typical $f_{MCGOUTCLK}$ = 21 MHz immediately after reset.
FEE (FLL engaged external)	$(f_{ext} / FLL\_R) \times F$	f_{ext} / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz
FBE (FLL bypassed external)	OSCCLK	OSCCLK / FLL_R must be in the range of 31.25 kHz to 39.0625 kHz
FBI (FLL bypassed internal)	MCGIRCLK	Selectable between slow and fast IRC
BLPI (Bypassed low power internal)	MCGIRCLK	Selectable between slow and fast IRC
BLPE (Bypassed low power external)	OSCCLK	

1. FLL_R is the reference divider selected by the C1[FRDIV] bits, F is the FLL factor selected by C4[DRST_DRS] and C4[DMX32] bits .

This section will include several mode switching examples, using an MHz external crystal..

Chapter 29
Oscillator (OSC)

29.1 Introduction

NOTE

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

The OSC module is a crystal oscillator. The module, in conjunction with an external crystal or resonator, generates a reference clock for the MCU.

29.2 Features and Modes

Key features of the module are listed here.

- Supports 32 kHz crystals (Low Range mode)
- Supports 3–8 MHz, 8–32 MHz crystals and resonators (High Range mode)
- Automatic Gain Control (AGC) to optimize power consumption in high frequency ranges 3–8 MHz, 8–32 MHz using low-power mode
- High gain option in frequency ranges: 32 kHz, 3–8 MHz, and 8–32 MHz
- Voltage and frequency filtering to guarantee clock frequency and stability
- Optionally external input bypass clock from EXTAL signal directly
- One clock for MCU clock system
- Two clocks for on-chip peripherals that can work in Stop modes

Functional Description describes the module's operation in more detail.

29.3 Block Diagram

The OSC module uses a crystal or resonator to generate three filtered oscillator clock signals.Three clocks are output from OSC module: OSCCLK for MCU system, OSCERCLK for on-chip peripherals, and OSC32KCLK. The OSCCLK can only work in run mode. OSCERCLK and OSC32KCLK can work in low power modes. For the clock source assignments, refer to the clock distribution information of this MCU.

Refer to the chip configuration details for the external reference clock source in this MCU.

The figure found here shows the block diagram of the OSC module.

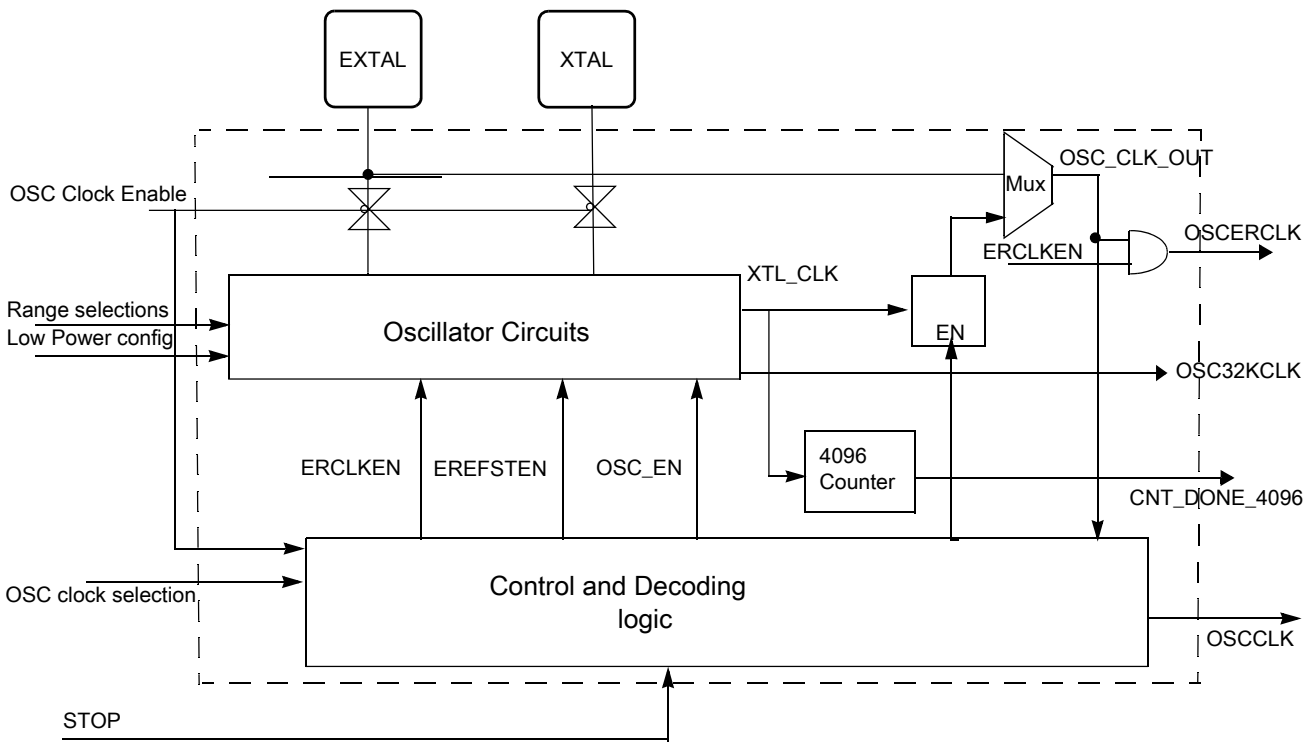


Figure 29-1. OSC Module Block Diagram

29.4 OSC Signal Descriptions

The table found here shows the user-accessible signals available for the OSC module.

Refer to signal multiplexing information for this MCU for more details.

Table 29-1. OSC Signal Descriptions

Signal	Description	I/O
EXTAL	External clock/Oscillator input	I
XTAL	Oscillator output	O

29.5 External Crystal / Resonator Connections

The connections for a crystal/resonator frequency reference are shown in the figures found here.

When using low-frequency, low-power mode, the only external component is the crystal or ceramic resonator itself. In the other oscillator modes, load capacitors (C_x, C_y) and feedback resistor (R_F) are required. The following table shows all possible connections.

Table 29-2. External Caystal/Resonator Connections

Oscillator Mode	Connections
Low-frequency (32 kHz), low-power	Connection 1[1]
Low-frequency (32 kHz), high-gain	Connection 2/Connection 3[2]
High-frequency (3~32 MHz), low-power	Connection 3[1]
High-frequency (3~32 MHz), high-gain	Connection 3

1. With the low-power mode, the oscillator has the internal feedback resistor R_F. Therefore, the feedback resistor must not be externally with the Connection 3.
2. When the load capacitors (Cx, Cy) are greater than 30 pF, use Connection 3.

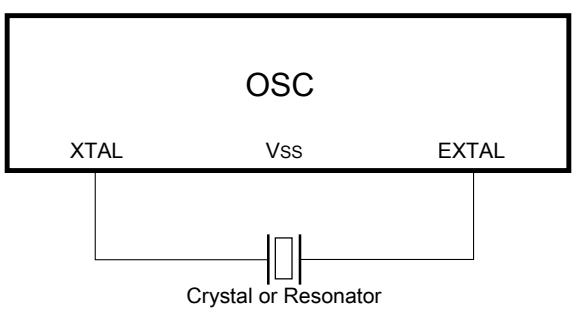


Figure 29-2. Crystal/Ceramic Resonator Connections - Connection 1

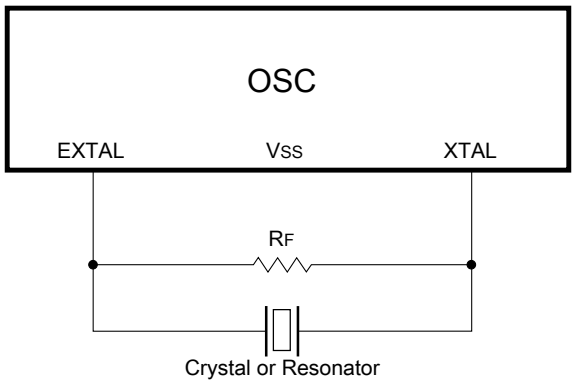


Figure 29-3. Crystal/Ceramic Resonator Connections - Connection 2

NOTE

Connection 1 and Connection 2 should use internal capacitors as the load of the oscillator by configuring the CR[SCxP] bits.

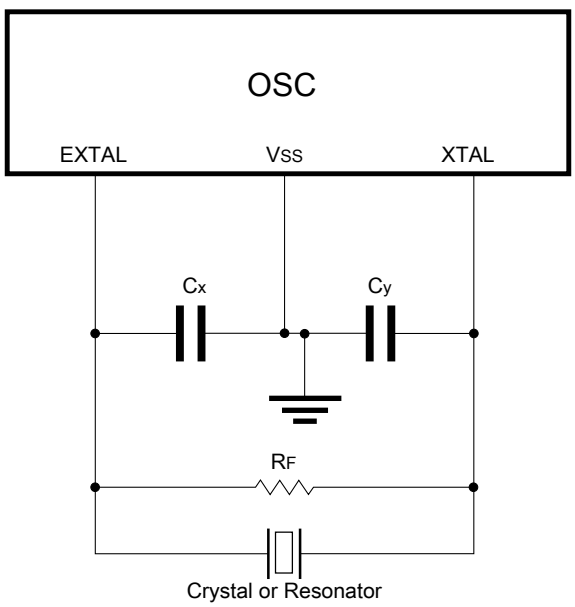


Figure 29-4. Crystal/Ceramic Resonator Connections - Connection 3

29.6 External Clock Connections

In external clock mode, the pins can be connected as shown in the figure found here.

NOTE

XTAL can be used as a GPIO when the GPIO alternate function is configured for it.

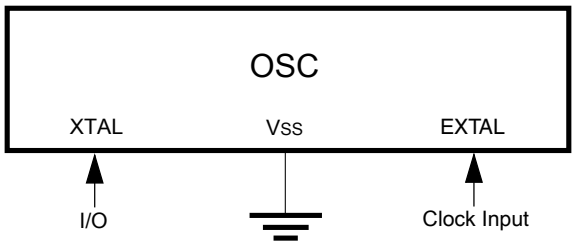


Figure 29-5. External Clock Connections

29.7 Memory Map/Register Definitions

Some oscillator module register bits are typically incorporated into other peripherals such as MCG or SIM.

29.7.1 OSC Memory Map/Register Definition

OSC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_5000	OSC Control Register (OSC_CR)	8	R/W	00h	29.7.1.1/ 573

29.7.1.1 OSC Control Register (OSC_CR)

NOTE

After OSC is enabled and starts generating the clocks, the configurations such as low power and frequency range, must not be changed.

Address: 4006_5000h base + 0h offset = 4006_5000h

Bit	7	6	5	4	3	2	1	0
Read	ERCLKEN	0	EREFSTEN	0	SC2P	SC4P	SC8P	SC16P
Write								
Reset	0	0	0	0	0	0	0	0

OSC_CR field descriptions

Field	Description
7 ERCLKEN	External Reference Enable Enables external reference clock (OSCERCLK) . 0 External reference clock is inactive. 1 External reference clock is enabled.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 EREFSTEN	External Reference Stop Enable Controls whether or not the external reference clock (OSCERCLK) remains enabled when MCU enters Stop mode. 0 External reference clock is disabled in Stop mode. 1 External reference clock stays enabled in Stop mode if ERCLKEN is set before entering Stop mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 SC2P	Oscillator 2 pF Capacitor Load Configure Configures the oscillator load. 0 Disable the selection. 1 Add 2 pF capacitor to the oscillator load.
2 SC4P	Oscillator 4 pF Capacitor Load Configure Configures the oscillator load. 0 Disable the selection. 1 Add 4 pF capacitor to the oscillator load.
1 SC8P	Oscillator 8 pF Capacitor Load Configure Configures the oscillator load. 0 Disable the selection. 1 Add 8 pF capacitor to the oscillator load.
0 SC16P	Oscillator 16 pF Capacitor Load Configure Configures the oscillator load. 0 Disable the selection. 1 Add 16 pF capacitor to the oscillator load.

29.8 Functional Description

Functional details of the module can be found here.

29.8.1 OSC module states

The states of the OSC module are shown in the following figure. The states and their transitions between each other are described in this section.

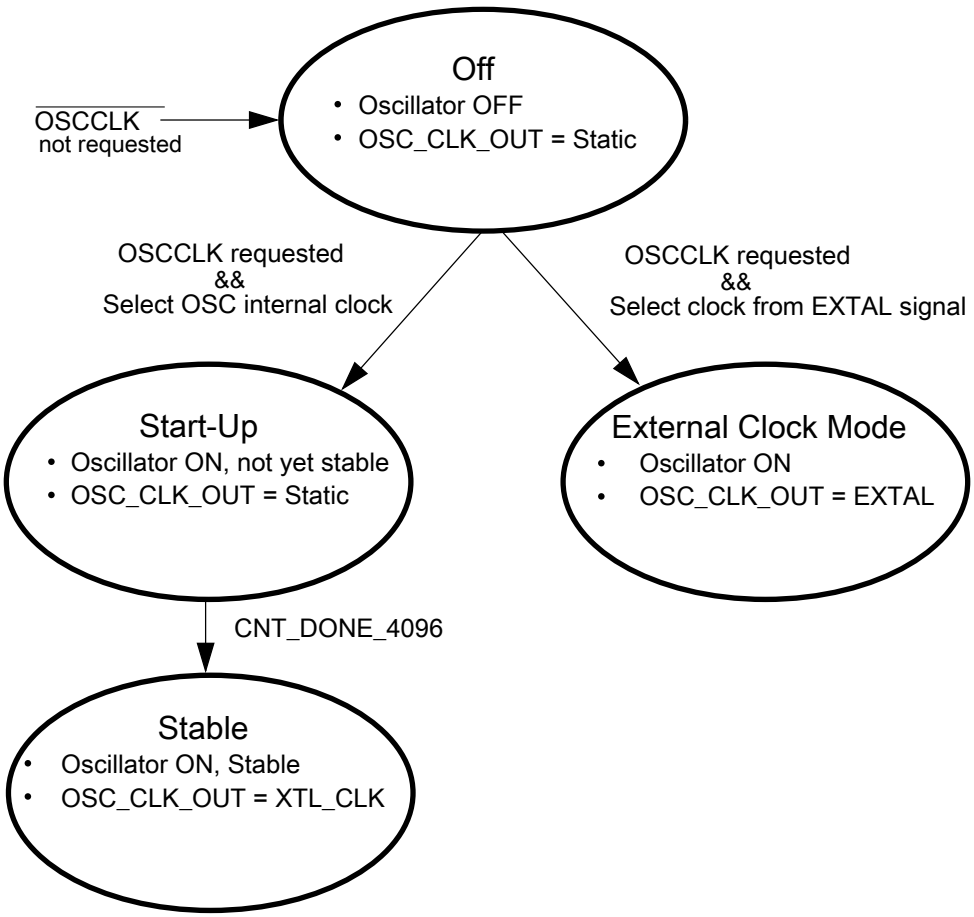


Figure 29-6. OSC Module state diagram

NOTE

XTL_CLK is the clock generated internally from OSC circuits.

29.8.1.1 Off

The OSC enters the Off state when the system does not require OSC clocks. Upon entering this state, XTL_CLK is static unless OSC is configured to select the clock from the EXTAL pad by clearing the external reference clock selection bit. For details regarding the external reference clock source in this MCU, refer to the chip configuration details. The EXTAL and XTAL pins are also decoupled from all other oscillator circuitry in this state. The OSC module circuitry is configured to draw minimal current.

29.8.1.2 Oscillator startup

The OSC enters startup state when it is configured to generate clocks (internally the OSC_EN transitions high) using the internal oscillator circuits by setting the external reference clock selection bit. In this state, the OSC module is enabled and oscillations are starting up, but have not yet stabilized. When the oscillation amplitude becomes large enough to pass through the input buffer, XTL_CLK begins clocking the counter. When the counter reaches 4096 cycles of XTL_CLK, the oscillator is considered stable and XTL_CLK is passed to the output clock OSC_CLK_OUT.

29.8.1.3 Oscillator Stable

The OSC enters stable state when it is configured to generate clocks (internally the OSC_EN transitions high) using the internal oscillator circuits by setting the external reference clock selection bit and the counter reaches 4096 cycles of XTL_CLK (when CNT_DONE_4096 is high). In this state, the OSC module is producing a stable output clock on OSC_CLK_OUT. Its frequency is determined by the external components being used.

29.8.1.4 External Clock mode

The OSC enters external clock state when it is enabled and external reference clock selection bit is cleared. For details regarding external reference clock source in this MCU, see the chip configuration details. In this state, the OSC module is set to buffer (with hysteresis) a clock from EXTAL onto the OSC_CLK_OUT. Its frequency is determined by the external clock being supplied.

29.8.2 OSC module modes

The OSC is a pierce-type oscillator that supports external crystals or resonators operating over the frequency ranges shown in Table 29-3. These modes assume the following conditions: OSC is enabled to generate clocks (OSC_EN=1), configured to generate clocks internally (MCG_C2[EREFS] = 1), and some or one of the other peripherals (MCG, Timer, and so on) is configured to use the oscillator output clock (OSC_CLK_OUT).

Table 29-3. Oscillator modes

Mode	Frequency Range
Low-frequency, high-gain	f_{osc_lo} (32.768 kHz) up to f_{osc_lo} (39.0625 kHz)
High-frequency mode1, high-gain	$f_{osc\_hi\_1}$ (3 MHz) up to $f_{osc\_hi\_1}$ (8 MHz)
High-frequency mode1, low-power	
High-frequency mode2, high-gain	$f_{osc\_hi\_2}$ (8 MHz) up to $f_{osc\_hi\_2}$ (32 MHz)
High-frequency mode2, low-power	

NOTE

For information about low power modes of operation used in this chip and their alignment with some OSC modes, see the chip's Power Management details.

29.8.2.1 Low-Frequency, High-Gain Mode

In Low-frequency, high-gain mode, the oscillator uses a simple inverter-style amplifier. The gain is set to achieve rail-to-rail oscillation amplitudes.

The oscillator input buffer in this mode is single-ended. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels. In this mode, the internal capacitors could be used.

29.8.2.2 Low-Frequency, Low-Power Mode

In low-frequency, low-power mode, the oscillator uses a gain control loop to minimize power consumption. As the oscillation amplitude increases, the amplifier current is reduced. This continues until a desired amplitude is achieved at steady-state. This mode provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels. In this mode, the internal capacitors could be used, the internal feedback resistor is connected, and no external resistor should be used.

In this mode, the amplifier inputs, gain-control input, and input buffer input are all capacitively coupled for leakage tolerance (not sensitive to the DC level of EXTAL).

Also in this mode, all external components except for the resonator itself are integrated, which includes the load capacitors and feeback resistor that biases EXTAL.

29.8.2.3 High-Frequency, High-Gain Mode

In high-frequency, high-gain mode, the oscillator uses a simple inverter-style amplifier. The gain is set to achieve rail-to-rail oscillation amplitudes. This mode provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels.

29.8.2.4 High-Frequency, Low-Power Mode

In high-frequency, low-power mode, the oscillator uses a gain control loop to minimize power consumption. As the oscillation amplitude increases, the amplifier current is reduced. This continues until a desired amplitude is achieved at steady-state. In this mode, no external resistor should be used.

The oscillator input buffer in this mode is differential. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels.

29.8.3 Counter

The oscillator output clock (OSC_CLK_OUT) is gated off until the counter has detected 4096 cycles of its input clock (XTL_CLK). After 4096 cycles are completed, the counter passes XTL_CLK onto OSC_CLK_OUT. This counting timeout is used to guarantee output clock stability.

29.8.4 Reference clock pin requirements

The OSC module requires use of both the EXTAL and XTAL pins to generate an output clock in Oscillator mode, but requires only the EXTAL pin in External clock mode. The EXTAL and XTAL pins are available for I/O. For the implementation of these pins on this device, refer to the Signal Multiplexing chapter.

29.9 Reset

There is no reset state associated with the OSC module. The counter logic is reset when the OSC is not configured to generate clocks.

There are no sources of reset requests for the OSC module.

29.10 Low power modes operation

When the MCU enters Stop modes, the OSC is functional depending on CR[ERCLKEN] and CR[EREFSETN] bit settings. If both these bits are set, the OSC is in operation.

After waking up from Very Low Leakage Stop (VLLSx) modes, all OSC register bits are reset and initialization is required through software.

29.11 Interrupts

The OSC module does not generate any interrupts.

Chapter 30
Flash Memory Controller (FMC)

30.1 Introduction

The Flash Memory Controller (FMC) is a memory acceleration unit. A list of features provided by the FMC can be found here.

- an interface between bus masters and the 64-bit program flash memory.
- a buffer and a cache that can accelerate program flash memory data transfers.

30.1.1 Overview

The Flash Memory Controller manages the interface between bus masters and the 64-bit program flash memory. The FMC receives status information detailing the configuration of the flash memory and uses this information to ensure a proper interface. The FMC supports 8-bit, 16-bit, and 32-bit read operations from the program flash memory. A write operation to program flash memory results in a bus error.

In addition, the FMC provides two separate mechanisms for accelerating the interface between bus masters and program flash memory. A 64-bit speculation buffer can prefetch the next 64-bit flash memory location, and a 4-way, 4-set program flash memory cache can store previously accessed program flash memory data for quick access times.

30.1.2 Features

The features of FMC module include:

- Interface between bus masters and the 64-bit program flash memory:
  - 8-bit, 16-bit, and 32-bit read operations to nonvolatile flash memory.
- Acceleration of data transfer from the program flash memory to the device:

- 64-bit prefetch speculation buffer for program flash accesses with controls for instruction/data access
- 4-way, 4-set, 64-bit line size program flash memory cache for a total of sixteen 64-bit entries with invalidation control

30.2 Modes of operation

The FMC operates only when a bus master accesses the program flash memory.

In terms of chip power modes:

- The FMC operates only in Run and Wait modes, including VLPR and VLPW modes.
- For any power mode where the program flash memory cannot be accessed, the FMC is disabled.

30.3 External signal description

The FMC has no external (off-chip) signals.

30.4 Memory map and register descriptions

The MCM's programming model provides control and configuration of the FMC's features.

For details, see the description of the MCM's Platform Control Register (PLACR).

30.5 Functional description

The FMC is a flash acceleration unit with flexible buffers for user configuration.

Besides managing the interface between bus masters and the program flash memory, the FMC can be used to customize the program flash memory cache and buffer to provide single-cycle system clock data access times. Whenever a hit occurs for the prefetch speculation buffer or the cache (when enabled), the requested data is transferred within a single system clock.

Upon system reset, the FMC is configured as follows:

- Flash cache is enabled.
- Instruction speculation and caching are enabled.

- Data speculation is disabled.
- Data caching is enabled.

Though the default configuration provides flash acceleration, advanced users may desire to customize the FMC buffer configurations to maximize throughput for their use cases. For example, the user may adjust the controls to enable buffering per access type (data or instruction).

NOTE

When reconfiguring the FMC, do not program the control and configuration inputs to the FMC while the program flash memory is being accessed. Instead, change them with a routine executing from RAM in supervisor mode.

Chapter 31
Flash Memory Module (FTFA)

31.1 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

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.

31.1.1 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.

31.1.1.1 Program Flash Memory Features

- Sector size of 2 KB
- Program flash protection scheme prevents accidental program or erase of stored data
- Program flash access control scheme prevents unauthorized access to selected code segments
- Automated, built-in, program and erase algorithms with verify

31.1.1.2 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

31.1.2 Block Diagram

The block diagram of the flash memory module is shown in the following figure.

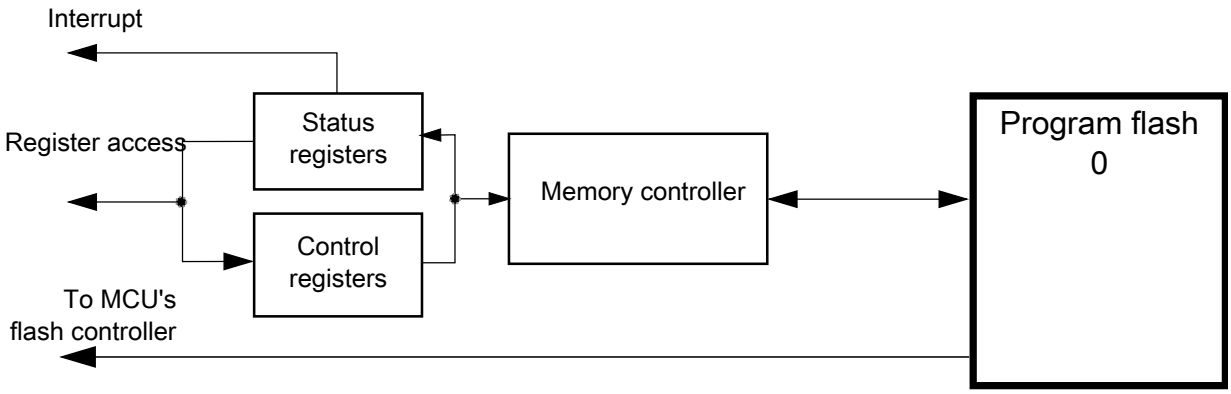


Figure 31-1. Flash Block Diagram

31.1.3 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.

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.

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.

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.

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.

31.2 External Signal Description

The flash memory module contains no signals that connect off-chip.

31.3 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.

31.3.1 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 Offset 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	Reserved
0x0_040E	1	Reserved
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).

31.3.2 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.

The program flash IFR is located within the program flash 0 memory block .

Address Range	Size (Bytes)	Field Description
0x00 – 0x9F	160	Reserved
0xA0 – 0xA3	4	Program Once XACCH-1 Field (index = 0x10)
0xA4 – 0xA7	4	Program Once XACCL-1 Field (index = 0x10)
0xA8 – 0xAB	4	Program Once XACCH-2 Field (index = 0x11)
0xAC – 0xAF	4	Program Once XACCL-2 Field (index = 0x11)
0xB0 – 0xB3	4	Program Once SACCH-1 Field (index = 0x12)
0xB4 – 0xB7	4	Program Once SACCL-1 Field (index = 0x12)

Table continues on the next page...

Address Range	Size (Bytes)	Field Description
0xB8 – 0xBB	4	Program Once SACCH-2 Field (index = 0x13)
0xBC – 0xBF	4	Program Once SACCL-2 Field (index = 0x13)
0xC0 – 0xFF	64	Program Once ID Field (index = 0x00 - 0x0F)

31.3.2.1 Program Once Field

The Program Once Field in the program flash IFR provides 96 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 or 8-Byte records using the Read Once and Program Once commands (see Read Once Command and Program Once Command).

31.3.3 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).

FTFA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_0000	Flash Status Register (FTFA_FSTAT)	8	R/W	00h	31.3.3.1/ 592
4002_0001	Flash Configuration Register (FTFA_FCNFG)	8	R/W	00h	31.3.3.2/ 593

Table continues on the next page...

FTFA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_0002	Flash Security Register (FTFA_FSEC)	8	R	Undefined	31.3.3.3/ 595
4002_0003	Flash Option Register (FTFA_FOPT)	8	R	Undefined	31.3.3.4/ 596
4002_0004	Flash Common Command Object Registers (FTFA_FCCOB3)	8	R/W	00h	31.3.3.5/ 597
4002_0005	Flash Common Command Object Registers (FTFA_FCCOB2)	8	R/W	00h	31.3.3.5/ 597
4002_0006	Flash Common Command Object Registers (FTFA_FCCOB1)	8	R/W	00h	31.3.3.5/ 597
4002_0007	Flash Common Command Object Registers (FTFA_FCCOB0)	8	R/W	00h	31.3.3.5/ 597
4002_0008	Flash Common Command Object Registers (FTFA_FCCOB7)	8	R/W	00h	31.3.3.5/ 597
4002_0009	Flash Common Command Object Registers (FTFA_FCCOB6)	8	R/W	00h	31.3.3.5/ 597
4002_000A	Flash Common Command Object Registers (FTFA_FCCOB5)	8	R/W	00h	31.3.3.5/ 597
4002_000B	Flash Common Command Object Registers (FTFA_FCCOB4)	8	R/W	00h	31.3.3.5/ 597
4002_000C	Flash Common Command Object Registers (FTFA_FCCOBB)	8	R/W	00h	31.3.3.5/ 597
4002_000D	Flash Common Command Object Registers (FTFA_FCCOBA)	8	R/W	00h	31.3.3.5/ 597
4002_000E	Flash Common Command Object Registers (FTFA_FCCOB9)	8	R/W	00h	31.3.3.5/ 597
4002_000F	Flash Common Command Object Registers (FTFA_FCCOB8)	8	R/W	00h	31.3.3.5/ 597
4002_0010	Program Flash Protection Registers (FTFA_FPROT3)	8	R/W	Undefined	31.3.3.6/ 598
4002_0011	Program Flash Protection Registers (FTFA_FPROT2)	8	R/W	Undefined	31.3.3.6/ 598
4002_0012	Program Flash Protection Registers (FTFA_FPROT1)	8	R/W	Undefined	31.3.3.6/ 598
4002_0013	Program Flash Protection Registers (FTFA_FPROT0)	8	R/W	Undefined	31.3.3.6/ 598
4002_001C	Execute-only Access Registers (FTFA_XACCL3)	8	R	Undefined	31.3.3.7/ 599
4002_001D	Execute-only Access Registers (FTFA_XACCL2)	8	R	Undefined	31.3.3.7/ 599
4002_001E	Execute-only Access Registers (FTFA_XACCL1)	8	R	Undefined	31.3.3.7/ 599
4002_001F	Execute-only Access Registers (FTFA_XACCL0)	8	R	Undefined	31.3.3.7/ 599

Table continues on the next page...

FTFA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_0024	Supervisor-only Access Registers (FTFA_SACCL3)	8	R	Undefined	31.3.3.8/ 600
4002_0025	Supervisor-only Access Registers (FTFA_SACCL2)	8	R	Undefined	31.3.3.8/ 600
4002_0026	Supervisor-only Access Registers (FTFA_SACCL1)	8	R	Undefined	31.3.3.8/ 600
4002_0027	Supervisor-only Access Registers (FTFA_SACCL0)	8	R	Undefined	31.3.3.8/ 600
4002_0028	Flash Access Segment Size Register (FTFA_FACSS)	8	R	Undefined	31.3.3.9/ 601
4002_002B	Flash Access Segment Number Register (FTFA_FACSN)	8	R	Undefined	31.3.3.10/ 602

31.3.3.1 Flash Status Register (FTFA_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 until the flag is cleared (by writing a one to it).

Address: 4002_0000h base + 0h offset = 4002_0000h

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

FTFA_FSTAT field descriptions

Field	Description
7 CCIF	Command Complete Interrupt Flag Indicates that a flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command, and CCIF stays low until command completion or command violation. 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...

FTFA_FSTAT field descriptions (continued)

Field	Description
	0 Flash command in progress 1 Flash command 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 ACCERR while CCIF is set. 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 memory during a command write sequence . While FPVIOL is set, the CCIF flag cannot be cleared to launch a command. The FPVIOL bit is cleared by writing a 1 to FPVIOL while CCIF is set. 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.

31.3.3.2 Flash Configuration Register (FTFA_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. The unassigned bits read as noted and are not writable.

Address: 4002_0000h base + 1h offset = 4002_0001h

Bit	7	6	5	4	3	2	1	0
Read	CCIE	RDCOLLIE	ERSAREQ	ERSSUSP	0	0	0	0
Write								
Reset	0	0	0	0	0	0	0	0

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

31.3.3.3 Flash Security Register (FTFA_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: 4002_0000h base + 2h offset = 4002_0002h

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.

FTFA_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 Enables and disables mass erase capability of the flash memory module at all times in all NVM modes. 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

Table continues on the next page...

FTFA_FSEC field descriptions (continued)

Field	Description
	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.

31.3.3.4 Flash Option Register (FTFA_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. However, the register is written to 0xFF if the contents of the flash nonvolatile option byte are 0x00.

Address: 4002_0000h base + 3h offset = 4002_0003h

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.

FTFA_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.

31.3.3.5 Flash Common Command Object Registers (FTFA_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: 4002_0000h 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

FTFA_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. 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

FTFA_FCCOB*n* field descriptions (continued)

Field	Description
	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).

31.3.3.6 Program Flash Protection Registers (FTFA_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 up to 32 protectable regions. Each bit protects a 1/32 region of the program flash memory except for memory configurations with less than 64 KB of program flash where each assigned bit protects 2 KB . For configurations with 48 KB of program flash memory or less, FPROT0 is not used. For configurations with 32 KB of program flash memory or less, FPROT1 is not used. For configurations with 16 KB of program flash memory, FPROT2 is not used. 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: 4002_0000h 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.

FTFA_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. 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. **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. Each bit in the 32-bit protection register represents 1/32 of the total program flash except for memory configurations with less than 64 KB of program flash where each assigned bit protects 2 KB . 0 Program flash region is protected. 1 Program flash region is not protected

31.3.3.7 Execute-only Access Registers (FTFA_XACC*n*)

The XACC registers define which program flash segments are restricted to data read or execute only or both data and instruction fetches.

The four XACC registers allow up to 32 restricted segments of equal memory size.

Execute-only access register	Program flash execute-only access bits
XACCL0	XA[31:24]
XACCL1	XA[23:16]
XACCL2	XA[15:8]
XACCL3	XA[7:0]

During the reset sequence, the XACC registers are loaded with the logical AND of Program Flash IFR addresses A and B as indicated in the following table.

Execute-only access register	Program Flash IFR address A	Program Flash IFR address B
XACCL0	0xA7	0xAF
XACCL1	0xA6	0xAE
XACCL2	0xA5	0xAD
XACCL3	0xA4	0xAC

Use the Program Once command to program the execute-only access control fields that are loaded during the reset sequence.

Address: 4002_0000h base + 1Ch offset + (1d × i), where i=0d to 3d

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

* Notes:
- x = Undefined at reset.

FTFA_XACC*n* field descriptions

Field	Description
XA	Execute-only access control 0 Associated segment is accessible in execute mode only (as an instruction fetch) 1 Associated segment is accessible as data or in execute mode

31.3.3.8 Supervisor-only Access Registers (FTFA_SACC*n*)

The SACC registers define which program flash segments are restricted to supervisor only or user and supervisor access.

The four SACC registers allow up to 32 restricted segments of equal memory size.

Supervisor-only access register	Program flash supervisor-only access bits
SACCL0	SA[31:24]
SACCL1	SA[23:16]
SACCL2	SA[15:8]
SACCL3	SA[7:0]

During the reset sequence, the SACC registers are loaded with the logical AND of Program Flash IFR addresses A and B as indicated in the following table.

Supervisor-only access register	Program Flash IFR address A	Program Flash IFR address B
SACCL0	0xB7	0xBF
SACCL1	0xB6	0xBE
SACCL2	0xB5	0xBD
SACCL3	0xB4	0xBC

Use the Program Once command to program the supervisor-only access control fields that are loaded during the reset sequence.

Address: 4002_0000h base + 24h offset + (1d × i), where i=0d to 3d

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

* Notes:
- x = Undefined at reset.

FTFA_SACC*n* field descriptions

Field	Description
SA	Supervisor-only access control 0 Associated segment is accessible in supervisor mode only 1 Associated segment is accessible in user or supervisor mode

31.3.3.9 Flash Access Segment Size Register (FTFA_FACSS)

The flash access segment size register determines which bits in the address are used to index into the SACC and XACC bitmaps to get the appropriate permission flags.

All bits in the register are read-only.

The contents of this register are loaded during the reset sequence.

Address: 4002_0000h base + 28h offset = 4002_0028h

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

* Notes:

- x = Undefined at reset.

FTFA_FACSS field descriptions

Field	Description
SGSIZE	Segment Size The segment size is a fixed value based on the available program flash size divided by NUMSG. Program Flash Size / Segment Size / Segment Size Encoding 64 KBytes / 2 KBytes / 0x3 128 KBytes / 4 KBytes / 0x4

31.3.3.10 Flash Access Segment Number Register (FTFA_FACSN)

The flash access segment number register provides the number of program flash segments that are available for XACC and SACC permissions.

All bits in the register are read-only.

The contents of this register are loaded during the reset sequence.

Address: 4002_0000h base + 2Bh offset = 4002_002Bh

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

* Notes:
- x = Undefined at reset.

FTFA_FACSN field descriptions

Field	Description
NUMSG	Number of Segments Indicator The NUMSG field indicates the number of equal-sized segments in the program flash. 0x20 Program flash memory is divided into 32 segments (64 Kbytes, 128 Kbytes)

31.4 Functional Description

The information found here describes functional details of the flash memory module.

31.4.1 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

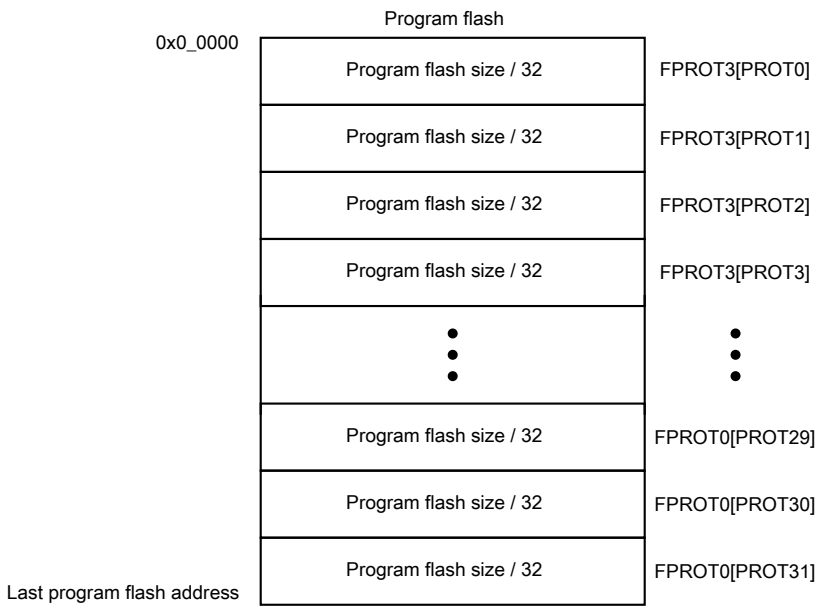


Figure 31-2. Program flash protection

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.

31.4.2 Flash Access Protection

Individual segments within the program flash memory can be designated for restricted access. Specific flash commands (Program Check, Program Longword, Erase Flash Sector) monitor FXACC contents to protect flash memory but the FSACC contents do not impact flash command operation.

See AN5112: Using the Kinetis Flash Execute-Only Access Control Feature for further details.

Access is controlled by the following registers:

- FTFA_XACC —
  - For 2^n program flash sizes 128KB or less, four registers control 32 segments of the program flash memory as shown in the following figure

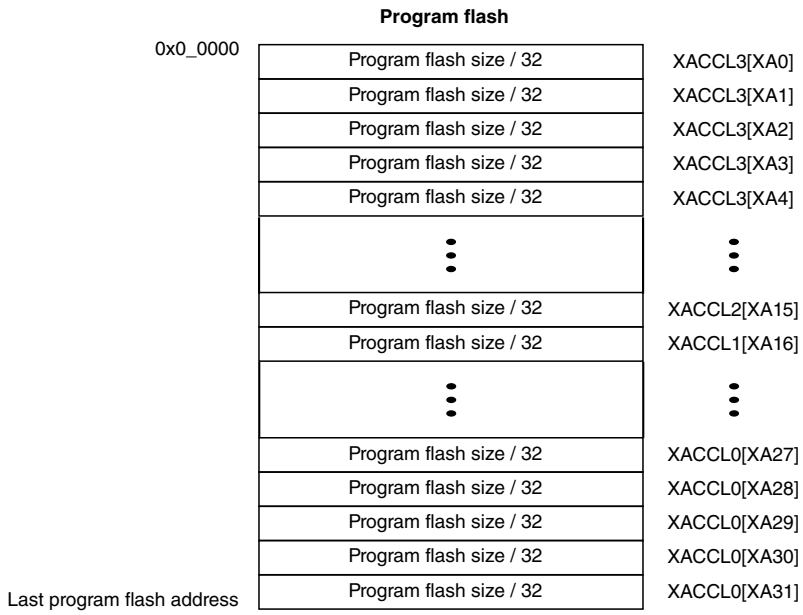


Figure 31-3. Program flash execute-only access control (64KB or 128KB of program flash)

- FTFA_SACC —
  - For 2^n program flash sizes 128KB or less, four registers control 32 segments of the program flash memory as shown in the following figure

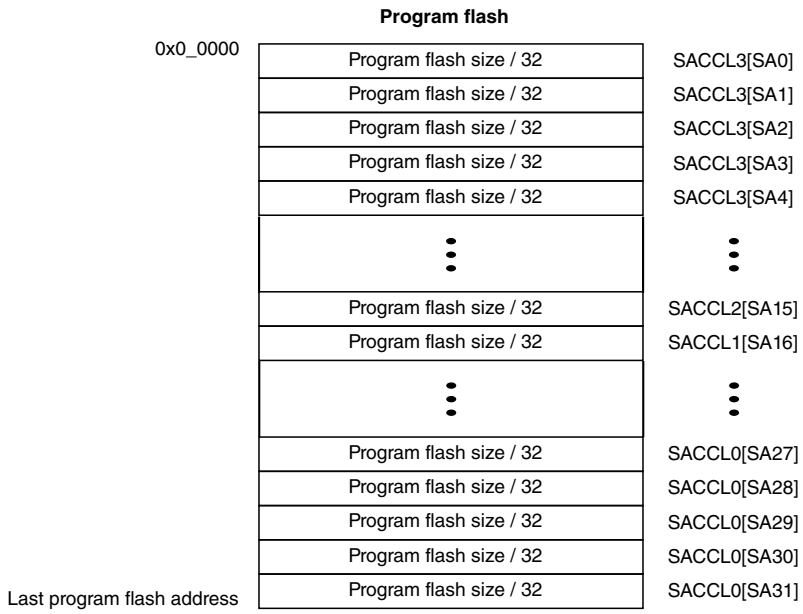


Figure 31-4. Program flash supervisor access control (64KB or 128KB of program flash)

31.4.3 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 31-1. 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.

31.4.4 Flash Operation in Low-Power Modes

31.4.4.1 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).

31.4.4.2 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.

31.4.5 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).

31.4.6 Read While Write (RWW)

The following simultaneous accesses are not allowed:

- Reading from program flash memory space while a flash command is active (CCIF=0).

31.4.7 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.

31.4.8 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

31.4.8.1 Command Write Sequence

Flash commands are specified using a command write sequence illustrated in Figure 31-5. 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.

31.4.8.1.1 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.

31.4.8.1.2 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.

31.4.8.1.3 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.

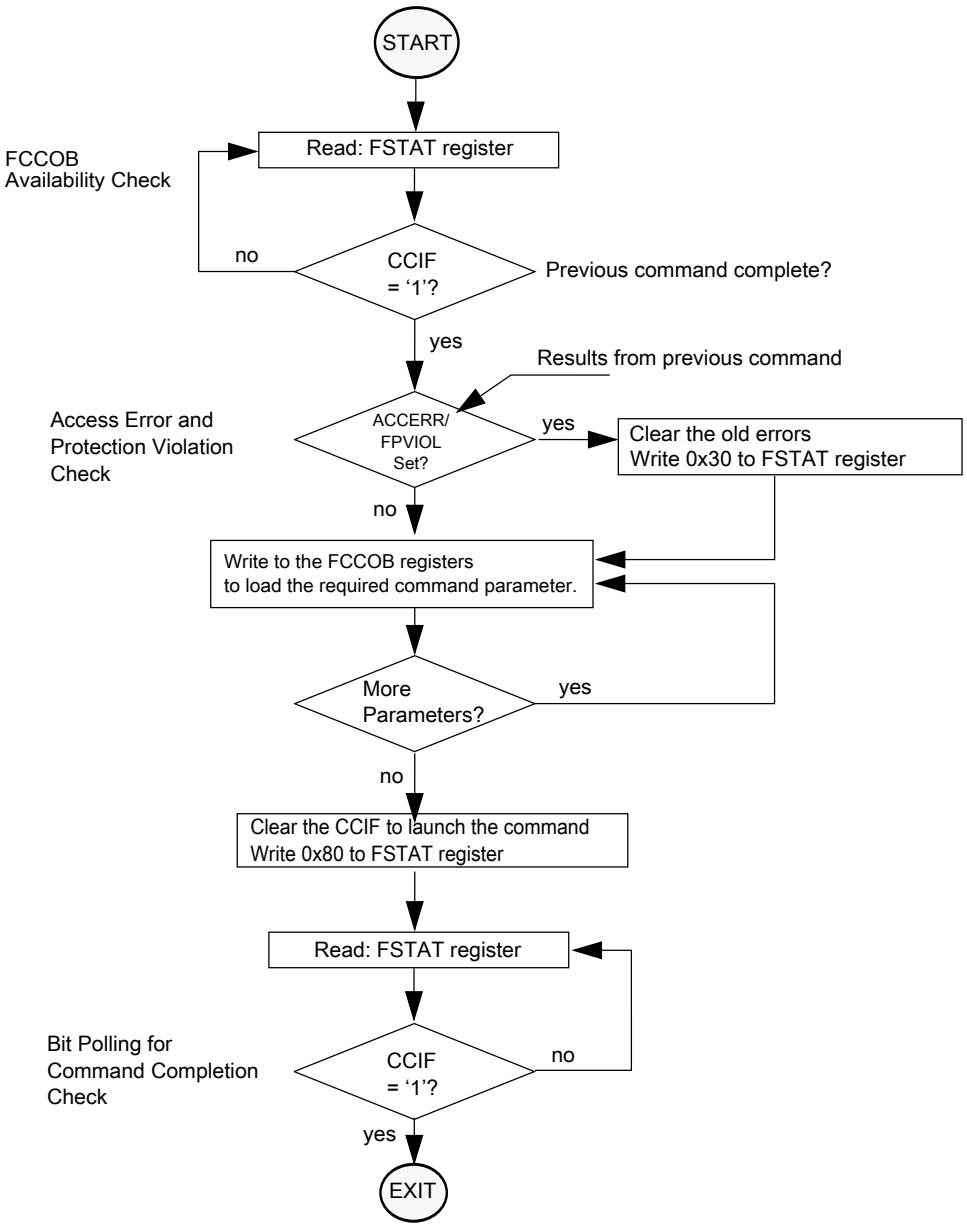


Figure 31-5. Generic flash command write sequence flowchart

31.4.8.2 Flash Commands

The following table summarizes the function of all flash commands.

FCMD	Command	Program flash	Function
0x01	Read 1s Section	×	Verify that a given number of program flash locations from a starting address are erased.
0x02	Program Check	×	Tests previously-programmed locations at margin read levels.
0x03	Read Resource	IFR, ID	Read 4 bytes from program flash IFR or version ID.
0x06	Program Longword	×	Program 4 bytes in a program flash block.
0x09	Erase Flash Sector	×	Erase all bytes in a program flash sector.
0x40	Read 1s All Blocks	×	Verify that the program flash block is erased then release MCU security.
0x41	Read Once	IFR	Read 4 bytes of a dedicated 64 byte field in the program flash 0 IFR.
0x43	Program Once	IFR	One-time program of 4 bytes of a dedicated 64-byte field in the program flash 0 IFR.
0x44	Erase All Blocks	×	Erase the program flash block, 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.
0x4A	Read 1s All Execute-only Segments	×	Verify that all program flash execute-only (XA) segments are erased then release flash access control.
0x4B	Erase All Execute-only Segments	×	Erase all program flash execute-only (XA) segments then release flash access control.

31.4.9 Margin Read Commands

The Read-1s commands (Read 1s All Blocks, Read 1s Section, Read 1s All Execute-only Segments) 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 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. Basic flash array reads 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.

31.4.10 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.

CAUTION

Flash data must be in the erased state before being programmed. Cumulative programming of bits (adding more zeros) is not allowed.

31.4.10.1 Read 1s Section Command

The Read 1s Section command checks if a section of program flash memory is erased to the specified read margin level. The Read 1s Section command defines the starting address and the number of phrases to be verified.

Table 31-2. Read 1s Section Command FCCOB Requirements

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).

Upon clearing CCIF to launch the Read 1s Section command, the flash memory module sets the read margin for 1s according to Table 31-3 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 31-3. 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 31-4. 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 aligned.	FSTAT[ACCERR]
The requested section crosses a Flash block boundary.	FSTAT[ACCERR]
The requested number of phrases is 0.	FSTAT[ACCERR]
Read-1s fails.	FSTAT[MGSTAT0]

31.4.10.2 Program Check Command

The Program Check command tests a previously programmed program flash longword to see if it reads correctly at the specified margin level.

Table 31-5. 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 31-6, 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 31-6. 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 31-7. 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]
Flash address is located in an XA controlled segment and the Erase All Blocks or the Read 1s All Blocks command has not successfully completed since the last reset	FSTAT[FPVIOL]
Either of the margin reads does not match the expected data	FSTAT[MGSTAT0]

31.4.10.3 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 and the Version ID field. Each resource is assigned a select code as shown in Table 31-9.

Table 31-8. 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 31-9)

1. Must be longword aligned (Flash address [1:0] = 00).

Table 31-9. Read Resource Select Codes

Resource Select Code	Description	Resource Size	Local Address Range
0x00	Program Flash 0 IFR	256 Bytes	0x00_0000–0x00_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 31-10. 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]

31.4.10.4 Program Longword Command

The Program Longword command programs four previously-erased bytes in the program 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 31-11. 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 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 31-12. 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]

Table continues on the next page...

Table 31-12. Program Longword Command Error Handling (continued)

Error Condition	Error Bit
Flash address is located in an XA controlled segment and the Erase All Blocks or the Read 1s All Blocks command has not successfully completed since the last reset	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

31.4.10.5 Erase Flash Sector Command

The Erase Flash Sector operation erases all addresses in a flash sector.

Table 31-13. 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] in the flash sector to be erased

1. Must be phrase aligned (flash address [2:0] = 000).

After clearing CCIF to launch the Erase Flash Sector command, the flash memory module erases the selected program 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 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 31-6).

Table 31-14. 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 aligned	FSTAT[ACCERR]
The selected program flash sector is protected	FSTAT[FPVIOL]
The selected program flash sector is located in an XA controlled segment and the Erase All Blocks or the Read 1s All Blocks command has not successfully completed since the last reset	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.

31.4.10.5.1 Suspending an Erase Flash Sector Operation

To suspend an Erase Flash Sector operation set the FCNFG[ERSSUSP] bit when CCIF, ACCERR, and FPVIOL are 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.

31.4.10.5.2 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 of 4.3 msec 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.

31.4.10.5.3 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.

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.

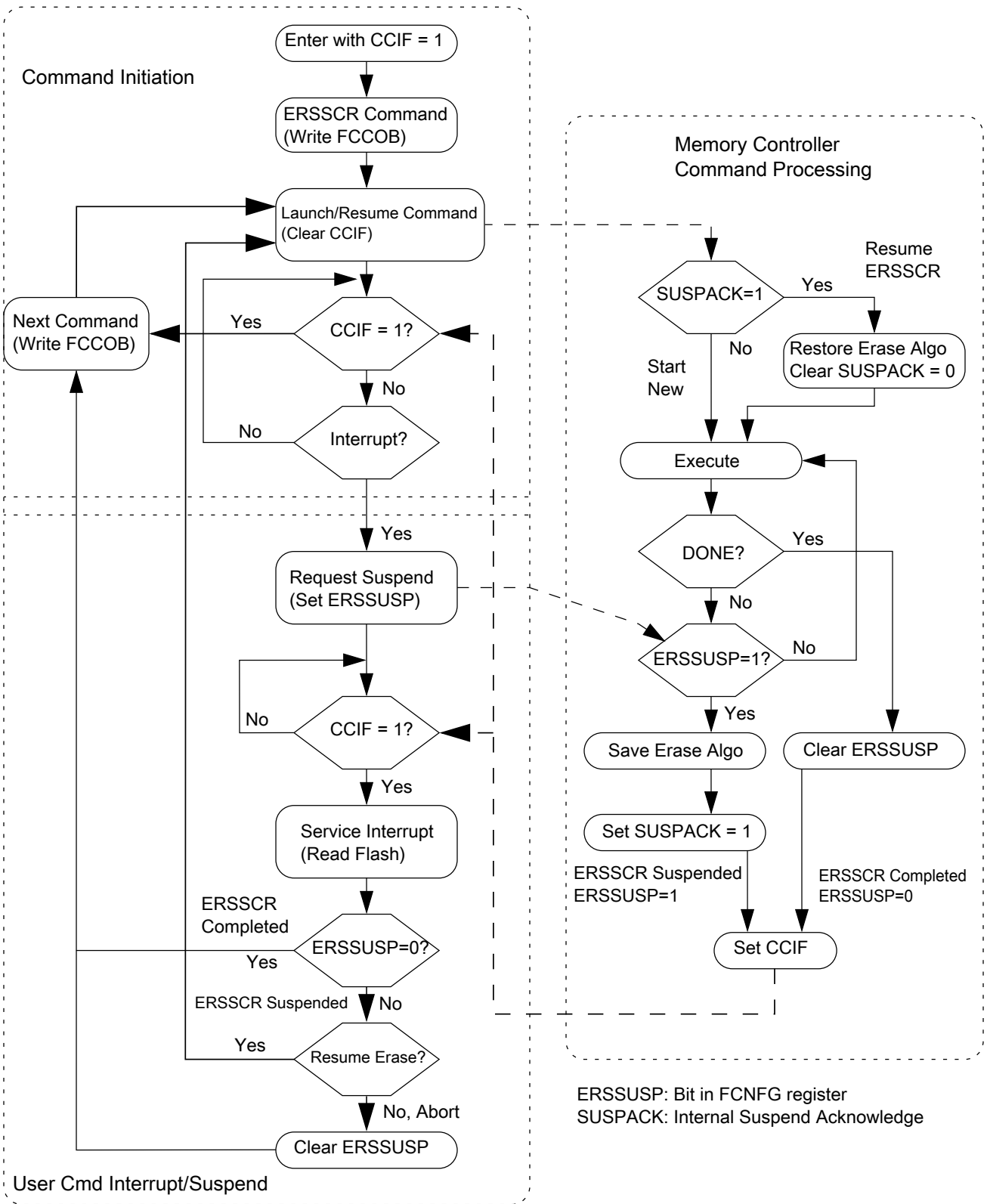


Figure 31-6. Suspend and Resume of Erase Flash Sector Operation

31.4.10.6 Read 1s All Blocks Command

The Read 1s All Blocks command checks if the program flash blocks 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 31-15. 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 31-16,
- checks the contents of the program flash are in the erased state.

If the flash memory module confirms that these memory resources are erased, access control is disabled and 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 CCIF flag sets after the Read 1s All Blocks operation has completed.

Table 31-16. 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 31-17. Read 1s All Blocks Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid margin choice is specified	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

31.4.10.7 Read Once Command

The Read Once command provides read access to special 96-byte fields located in the program flash 0 IFR (see Program Flash IFR Map and Program Once Field). Access to the Program Once ID field is via 16 records (index values 0x00 - 0x0F), each 4 bytes long. Access to the Program Once XACC and SACC fields are via 4 records (index values 0x10 - 0x13), each of which is 8 bytes long. These fields are programmed using the Program Once command described in Program Once Command.

Table 31-18. Read Once Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x41 (RDONCE)
1	Program Once record index (0x00 - 0x13)
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
8	Program Once byte 4 value (index 0x10 - 0x13)
9	Program Once byte 5 value (index 0x10 - 0x13)
10	Program Once byte 6 value (index 0x10 - 0x13)
11	Program Once byte 7 value (index 0x10 - 0x13)

After clearing CCIF to launch the Read Once command, a 4-byte or 8-byte Program Once record is read 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 - 0x13. During execution of the Read Once command, any attempt to read addresses within the program flash block containing the selected record index returns invalid data. The Read Once command can be executed any number of times.

Table 31-19. 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]

31.4.10.8 Program Once Command

The Program Once command enables programming to special 96-byte fields in the program flash 0 IFR (see Program Flash IFR Map and Program Once Field). Access to the Program Once ID field is via 16 records (index values 0x00 - 0x0F), each 4 bytes long. Access to the Program Once XACC and SACC fields are via 4 records (index values 0x10 - 0x13), each of which is 8 bytes long. These records can be read using the Read Once command (see Read Once Command) or using the Read Resource command (see Read Resource Command). These records can be programmed only once since the program flash 0 IFR cannot be erased.

Table 31-20. Program Once Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x43 (PGMONCE)
1	Program Once record index (0x00 - 0x13)
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
8	Program Once byte 4 value (index 0x10 - 0x13)
9	Program Once byte 5 value (index 0x10 - 0x13)
10	Program Once byte 6 value (index 0x10 - 0x13)
11	Program Once byte 7 value (index 0x10 - 0x13)

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.

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 - 0x13. During execution of the Program Once command, any attempt to read addresses within the program flash block containing the selected record index returns invalid data.

Table 31-21. 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]

Table continues on the next page...

Table 31-21. Program Once Command Error Handling (continued)

Error Condition	Error Bit
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

1. If a Program Once record is initially programmed to 0xFFFF_FFFF (0xFFFF_FFFF_FFFF_FFFF for index 0x10 - 0x13), the Program Once command is allowed to execute again on that same record.

31.4.10.9 Erase All Blocks Command

The Erase All Blocks operation erases all flash memory, verifies all memory contents, and releases MCU security.

Table 31-22. 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, then verifies that all are erased.

If the flash memory module verifies that all flash memories were properly erased, access control is disabled and security is released by setting the FSEC[SEC] field to the unsecure state. The Erase All Blocks command aborts if any flash 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.

Access control determined by the contents of the FXACC registers will not block execution of the Erase All Blocks command. While most Flash memory will be erased, the program flash IFR space containing the Program Once XACC and SACC fields will not be erased and, therefore, the contents of the Program Once XACC and SACC fields will not change. The contents of the FXACC and FSACC registers will not be impacted by the execution of the Erase All Blocks command. After completion of the Erase All Blocks command, access control is disabled until the next reset of the flash module or the Read 1s All Blocks command is executed and fails (FSTAT[MGSTAT0] is set).

Table 31-23. 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 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.

31.4.10.9.1 Triggering an Erase All External to the Flash Memory Module

The functionality of the Erase All Blocks 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 regardless of the protection settings. If the post-erase verify passes, access control determined by the contents of the FXACC registers is disabled and the routine then releases security by setting the FSEC[SEC] field register to the unsecure state. 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, except FPVIOL, as described in Erase All Blocks Command.

31.4.10.10 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 31-24. 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_0003
5	Key Byte 1	0x0_0002
6	Key Byte 2	0x0_0001
7	Key Byte 3	0x0_0000
8	Key Byte 4	0x0_0007
9	Key Byte 5	0x0_0006
A	Key Byte 6	0x0_0005

Table continues on the next page...

Table 31-24. Verify Backdoor Access Key Command FCCOB Requirements (continued)

FCCOB Number	FCCOB Contents [7:0]	Flash Configuration Field Offset Address
B	Key Byte 7	0x0_0004

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 31-25. Verify Backdoor Access Key Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
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]

31.4.10.11 Read 1s All Execute-only Segments Command

The Read 1s All Execute-only Segments command checks if the program flash execute-only segments defined by the FXACC registers have been erased to the specified read margin level, if applicable, and releases flash access control if the readout passes, i.e. all data reads as '1'.

Table 31-26. Read 1s All Execute-only Segments Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x4A (RD1XA)
1	Read-1 Margin Choice

After clearing CCIF to launch the Read 1s All Execute-only Segments command, the flash memory module :

- sets the read margin for 1s according to Table 31-27,
- checks the contents of the program flash execute-only segments are in the erased state.

If the flash memory module confirms that these segments are erased, flash access control is disabled until the next reset or, after programming any of the execute-only segments, the Read 1s All Execute-only Segments command is executed and fails with the FSTAT[MGSTAT0] bit set. If the read fails, i.e. all segments are not in the fully erased state, the FSTAT[MGSTAT0] bit is set.

The CCIF flag sets after the Read 1s All Execute-only Segments operation has completed.

Table 31-27. Margin Level Choices for Read 1s All Execute-only Segments

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 31-28. Read 1s All Execute-only Segments Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid margin choice is specified	FSTAT[ACCERR]
Sector size is larger than segment size	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

31.4.10.12 Erase All Execute-only Segments Command

The Erase All Execute-only Segments operation erases all program flash execute-only segments defined by the FXACC registers, verifies all segments are erased, and releases flash access control.

Table 31-29. Erase All Execute-only Segments Command FCCOB Requirements

FCCOB Number	FCCOB Contents [7:0]
0	0x4B (ERSXA)

After clearing CCIF to launch the Erase All Execute-only Segments command, the flash memory module erases all program flash execute-only segments, then verifies that all segments are erased.

If the flash memory module verifies that all segments were properly erased, flash access control is disabled until the next reset or, after programming any of the execute-only segments, the Read 1s All Execute-only Segments command is executed and fails with the FSTAT[MGSTAT0] bit set. The Erase All Execute-only Segments command aborts if any XA controlled segment is protected. If the erase-verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase All Execute-only Segments operation completes.

Access control determined by the contents of the FXACC registers will not block execution of the Erase All Execute-only Segments command. While all XA controlled segments will be erased, the program flash IFR space containing the Program Once XACC fields will not be erased and, therefore, the contents of the Program Once XACC fields will not change. The contents of the FXACC registers will not be impacted by the execution of the Erase All Execute-only Segments command.

Table 31-30. Erase All Execute-only Segments Command Error Handling

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Sector size is larger than segment size	FSTAT[ACCERR]
Any XA controlled segment in the program flash memory is protected	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

31.4.11 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 (FTFA_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 31-31. FSEC register fields

FSEC field	Description
KEYEN	Backdoor Key Access
MEEN	Mass Erase Capability
FSLACC	Factory Security Level Access
SEC	MCU security

31.4.11.1 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.

31.4.11.1.1 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.

31.4.12 Reset Sequence

On each system reset the flash memory module executes a sequence which establishes initial values for the flash block configuration parameters, FPROT, FOPT, FSEC, FXACC, FSACC, and FACNFG registers.

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 32
Cyclic Redundancy Check (CRC)

32.1 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 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.

32.1.1 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

32.1.2 Block diagram

The following is a block diagram of the CRC.

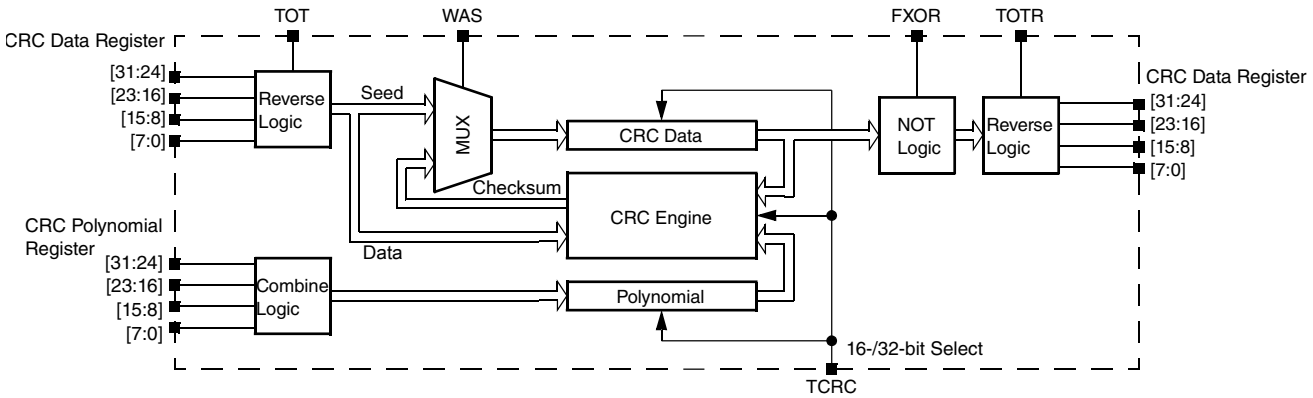


Figure 32-1. Programmable cyclic redundancy check (CRC) block diagram

32.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

32.1.3.1 Run mode

This is the basic mode of operation.

32.1.3.2 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.

32.2 Memory map and register descriptions

CRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_2000	CRC Data register (CRC_DATA)	32	R/W	FFFF_FFFFh	32.2.1/633
4003_2004	CRC Polynomial register (CRC_GPOLY)	32	R/W	0000_1021h	32.2.2/634
4003_2008	CRC Control register (CRC_CTRL)	32	R/W	0000_0000h	32.2.3/634

32.2.1 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.

Address: 4003_2000h base + 0h offset = 4003_2000h

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

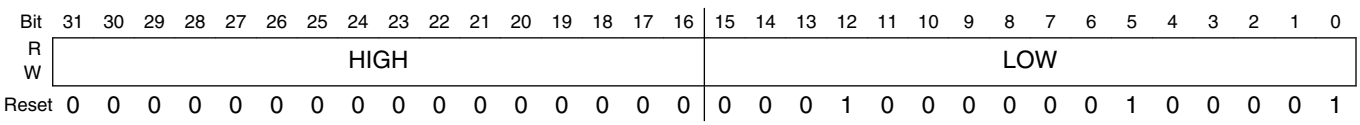
CRC_DATA field descriptions

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.
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.

32.2.2 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: 4003_2000h base + 4h offset = 4003_2004h



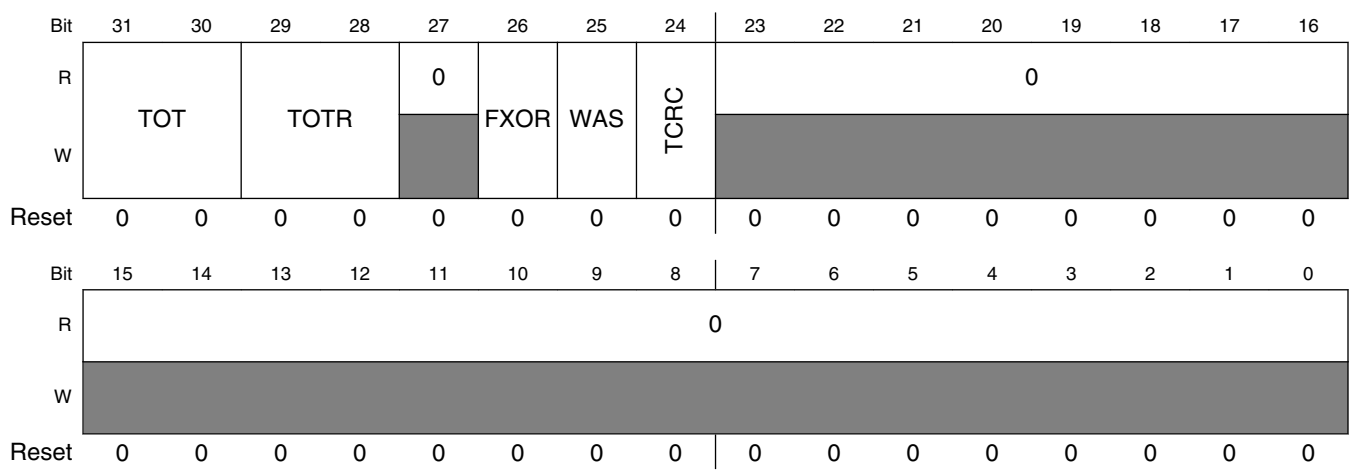
CRC_GPOLY field descriptions

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.

32.2.3 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: 4003_2000h base + 8h offset = 4003_2008h



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. 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.

32.3 Functional description

32.3.1 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.

32.3.2 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.

32.3.2.1 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.

32.3.2.2 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.

32.3.3 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.

32.3.3.1 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]}

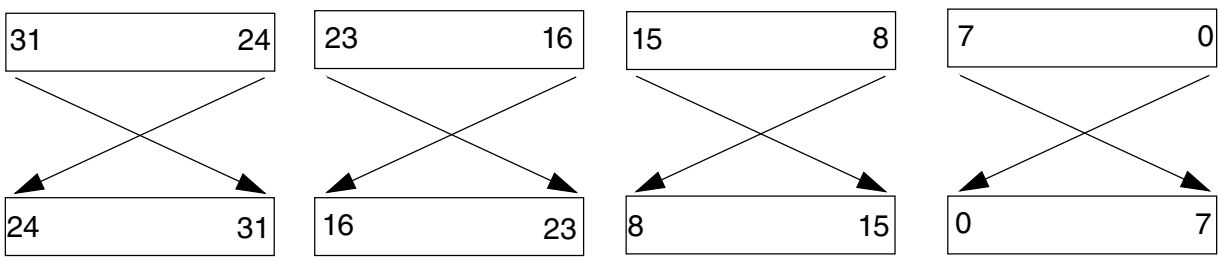


Figure 32-2. Transpose type 01

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]}

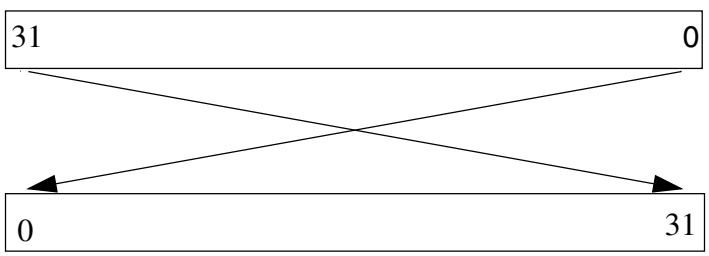


Figure 32-3. Transpose type 10

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]}

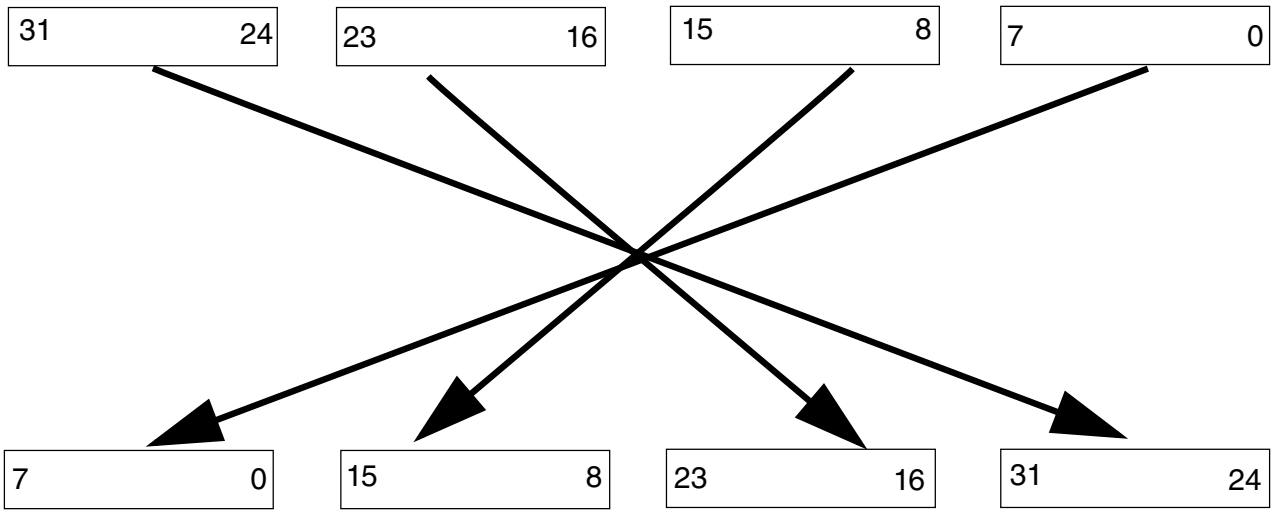


Figure 32-4. Transpose type 11

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.

32.3.4 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 33
Analog-to-Digital Converter (ADC)

33.1 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.

33.1.1 Features

Following are the features of the ADC module.

- Linear successive approximation algorithm with up to 16-bit resolution
- Up to four pairs of differential and 24 single-ended external analog inputs
- Output modes:
  - differential 16-bit, 13-bit, 11-bit, and 9-bit modes
  - single-ended 16-bit, 12-bit, 10-bit, and 8-bit modes
- Output format in 2's complement 16-bit sign extended for differential 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

33.1.2 Block diagram

The following figure is the ADC module block diagram.

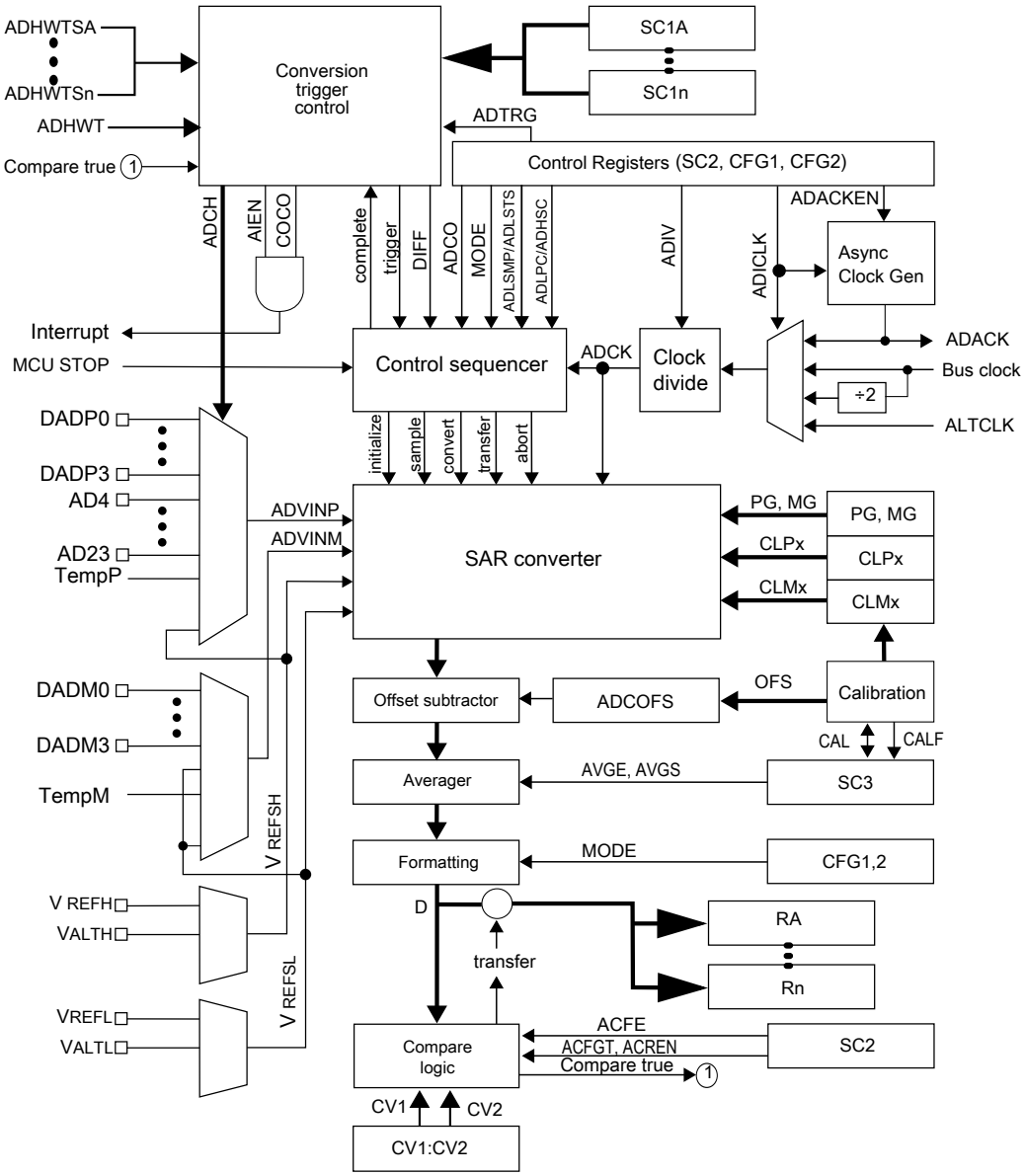


Figure 33-1. ADC block diagram

33.2 ADC signal descriptions

The ADC module supports up to 4 pairs of differential inputs and up to 24 single-ended inputs.

Each differential pair requires two inputs, DADPx and DADMx. The ADC also requires four supply/reference/ground connections.

NOTE

For the number of channels supported on this device as well as information regarding other chip-specific inputs into the ADC block, see the chip-specific ADC configuration information.

Table 33-1. ADC signal descriptions

Signal	Description	I/O
DADP3–DADP0	Differential Analog Channel Inputs	I
DADM3–DADM0	Differential Analog Channel Inputs	I
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

33.2.1 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.

33.2.2 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}.

33.2.3 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}.

33.2.4 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 when SC1n[DIFF] is low.

33.2.5 Differential Analog Channel Inputs (DADx)

The ADC module supports up to four differential analog channel inputs. Each differential analog input is a pair of external pins, DADPx and DADMx, referenced to each other to provide the most accurate analog to digital readings. A differential input is selected for conversion through SC1[ADCH] when SC1n[DIFF] is high. All DADPx inputs may be used as single-ended inputs if SC1n[DIFF] is low. In certain MCU configurations, some DADMx inputs may also be used as single-ended inputs if SC1n[DIFF] is low. For ADC connections specific to this device, see the chip-specific ADC information.

33.3 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
4003_B000	ADC Status and Control Registers 1 (ADC0_SC1A)	32	R/W	0000_001Fh	33.3.1/647
4003_B004	ADC Status and Control Registers 1 (ADC0_SC1B)	32	R/W	0000_001Fh	33.3.1/647
4003_B008	ADC Configuration Register 1 (ADC0_CFG1)	32	R/W	0000_0000h	33.3.2/651
4003_B00C	ADC Configuration Register 2 (ADC0_CFG2)	32	R/W	0000_0000h	33.3.3/653
4003_B010	ADC Data Result Register (ADC0_RA)	32	R	0000_0000h	33.3.4/654

Table continues on the next page...

ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_B014	ADC Data Result Register (ADC0_RB)	32	R	0000_0000h	33.3.4/654
4003_B018	Compare Value Registers (ADC0_CV1)	32	R/W	0000_0000h	33.3.5/655
4003_B01C	Compare Value Registers (ADC0_CV2)	32	R/W	0000_0000h	33.3.5/655
4003_B020	Status and Control Register 2 (ADC0_SC2)	32	R/W	0000_0000h	33.3.6/656
4003_B024	Status and Control Register 3 (ADC0_SC3)	32	R/W	0000_0000h	33.3.7/658
4003_B028	ADC Offset Correction Register (ADC0_OFS)	32	R/W	0000_0004h	33.3.8/659
4003_B02C	ADC Plus-Side Gain Register (ADC0_PG)	32	R/W	0000_8200h	33.3.9/660
4003_B030	ADC Minus-Side Gain Register (ADC0_MG)	32	R/W	0000_8200h	33.3.10/ 661
4003_B034	ADC Plus-Side General Calibration Value Register (ADC0_CLPD)	32	R/W	0000_000Ah	33.3.11/ 661
4003_B038	ADC Plus-Side General Calibration Value Register (ADC0_CLPS)	32	R/W	0000_0020h	33.3.12/ 662
4003_B03C	ADC Plus-Side General Calibration Value Register (ADC0_CLP4)	32	R/W	0000_0200h	33.3.13/ 662
4003_B040	ADC Plus-Side General Calibration Value Register (ADC0_CLP3)	32	R/W	0000_0100h	33.3.14/ 663
4003_B044	ADC Plus-Side General Calibration Value Register (ADC0_CLP2)	32	R/W	0000_0080h	33.3.15/ 663
4003_B048	ADC Plus-Side General Calibration Value Register (ADC0_CLP1)	32	R/W	0000_0040h	33.3.16/ 664
4003_B04C	ADC Plus-Side General Calibration Value Register (ADC0_CLP0)	32	R/W	0000_0020h	33.3.17/ 664
4003_B054	ADC Minus-Side General Calibration Value Register (ADC0_CLMD)	32	R/W	0000_000Ah	33.3.18/ 665
4003_B058	ADC Minus-Side General Calibration Value Register (ADC0_CLMS)	32	R/W	0000_0020h	33.3.19/ 665
4003_B05C	ADC Minus-Side General Calibration Value Register (ADC0_CLM4)	32	R/W	0000_0200h	33.3.20/ 666
4003_B060	ADC Minus-Side General Calibration Value Register (ADC0_CLM3)	32	R/W	0000_0100h	33.3.21/ 666
4003_B064	ADC Minus-Side General Calibration Value Register (ADC0_CLM2)	32	R/W	0000_0080h	33.3.22/ 667
4003_B068	ADC Minus-Side General Calibration Value Register (ADC0_CLM1)	32	R/W	0000_0040h	33.3.23/ 667
4003_B06C	ADC Minus-Side General Calibration Value Register (ADC0_CLM0)	32	R/W	0000_0020h	33.3.24/ 668
4003_C000	ADC Status and Control Registers 1 (ADC1_SC1A)	32	R/W	0000_001Fh	33.3.1/647
4003_C004	ADC Status and Control Registers 1 (ADC1_SC1B)	32	R/W	0000_001Fh	33.3.1/647
4003_C008	ADC Configuration Register 1 (ADC1_CFG1)	32	R/W	0000_0000h	33.3.2/651
4003_C00C	ADC Configuration Register 2 (ADC1_CFG2)	32	R/W	0000_0000h	33.3.3/653
4003_C010	ADC Data Result Register (ADC1_RA)	32	R	0000_0000h	33.3.4/654

Table continues on the next page...

ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_C014	ADC Data Result Register (ADC1_RB)	32	R	0000_0000h	33.3.4/654
4003_C018	Compare Value Registers (ADC1_CV1)	32	R/W	0000_0000h	33.3.5/655
4003_C01C	Compare Value Registers (ADC1_CV2)	32	R/W	0000_0000h	33.3.5/655
4003_C020	Status and Control Register 2 (ADC1_SC2)	32	R/W	0000_0000h	33.3.6/656
4003_C024	Status and Control Register 3 (ADC1_SC3)	32	R/W	0000_0000h	33.3.7/658
4003_C028	ADC Offset Correction Register (ADC1_OFS)	32	R/W	0000_0004h	33.3.8/659
4003_C02C	ADC Plus-Side Gain Register (ADC1_PG)	32	R/W	0000_8200h	33.3.9/660
4003_C030	ADC Minus-Side Gain Register (ADC1_MG)	32	R/W	0000_8200h	33.3.10/ 661
4003_C034	ADC Plus-Side General Calibration Value Register (ADC1_CLPD)	32	R/W	0000_000Ah	33.3.11/ 661
4003_C038	ADC Plus-Side General Calibration Value Register (ADC1_CLPS)	32	R/W	0000_0020h	33.3.12/ 662
4003_C03C	ADC Plus-Side General Calibration Value Register (ADC1_CLP4)	32	R/W	0000_0200h	33.3.13/ 662
4003_C040	ADC Plus-Side General Calibration Value Register (ADC1_CLP3)	32	R/W	0000_0100h	33.3.14/ 663
4003_C044	ADC Plus-Side General Calibration Value Register (ADC1_CLP2)	32	R/W	0000_0080h	33.3.15/ 663
4003_C048	ADC Plus-Side General Calibration Value Register (ADC1_CLP1)	32	R/W	0000_0040h	33.3.16/ 664
4003_C04C	ADC Plus-Side General Calibration Value Register (ADC1_CLP0)	32	R/W	0000_0020h	33.3.17/ 664
4003_C054	ADC Minus-Side General Calibration Value Register (ADC1_CLMD)	32	R/W	0000_000Ah	33.3.18/ 665
4003_C058	ADC Minus-Side General Calibration Value Register (ADC1_CLMS)	32	R/W	0000_0020h	33.3.19/ 665
4003_C05C	ADC Minus-Side General Calibration Value Register (ADC1_CLM4)	32	R/W	0000_0200h	33.3.20/ 666
4003_C060	ADC Minus-Side General Calibration Value Register (ADC1_CLM3)	32	R/W	0000_0100h	33.3.21/ 666
4003_C064	ADC Minus-Side General Calibration Value Register (ADC1_CLM2)	32	R/W	0000_0080h	33.3.22/ 667
4003_C068	ADC Minus-Side General Calibration Value Register (ADC1_CLM1)	32	R/W	0000_0040h	33.3.23/ 667
4003_C06C	ADC Minus-Side General Calibration Value Register (ADC1_CLM0)	32	R/W	0000_0020h	33.3.24/ 668

33.3.1 ADC Status and Control Registers 1 (ADC*x*_SC1*n*)

SC1A is used for both software and hardware trigger modes of operation.

To allow sequential conversions of the ADC to be triggered by internal peripherals, the ADC can have more than one status and control register: one for each conversion. The SC1B–SC1n registers indicate potentially multiple SC1 registers for use only in hardware trigger mode. See the chip configuration information about the number of SC1n registers specific to this device. The SC1n registers have identical fields, and are used in a "ping-pong" approach to control ADC operation.

At any one point in time, only one of the SC1n registers is actively controlling ADC conversions. Updating SC1A while SC1n is actively controlling a conversion is allowed, and vice-versa for any of the SC1n registers specific to this MCU.

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 (module disabled).

Writing any of the SC1n registers while that specific SC1n register is actively controlling a conversion aborts the current conversion. None of the SC1B-SC1n registers are used for software trigger operation and therefore writes to the SC1B–SC1n registers do not initiate a new conversion.

Address: Base address + 0h offset + (4d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								COCO	AIEN	DIFF	ADCH				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

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.
5 DIFF	Differential Mode Enable Configures the ADC to operate in differential mode. When enabled, this mode automatically selects from the differential channels, and changes the conversion algorithm and the number of cycles to complete a conversion. 0 Single-ended conversions and input channels are selected. 1 Differential conversions and input channels are selected.
ADCH	Input channel select Selects one of the input channels. The input channel decode depends on the value of DIFF. DAD0-DAD3 are associated with the input pin pairs DADPx and DADMx. **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 When DIFF=0, DADP0 is selected as input; when DIFF=1, DAD0 is selected as input. 00001 When DIFF=0, DADP1 is selected as input; when DIFF=1, DAD1 is selected as input. 00010 When DIFF=0, DADP2 is selected as input; when DIFF=1, DAD2 is selected as input. 00011 When DIFF=0, DADP3 is selected as input; when DIFF=1, DAD3 is selected as input. 00100 When DIFF=0, AD4 is selected as input; when DIFF=1, it is reserved. 00101 When DIFF=0, AD5 is selected as input; when DIFF=1, it is reserved. 00110 When DIFF=0, AD6 is selected as input; when DIFF=1, it is reserved. 00111 When DIFF=0, AD7 is selected as input; when DIFF=1, it is reserved.

Table continues on the next page...

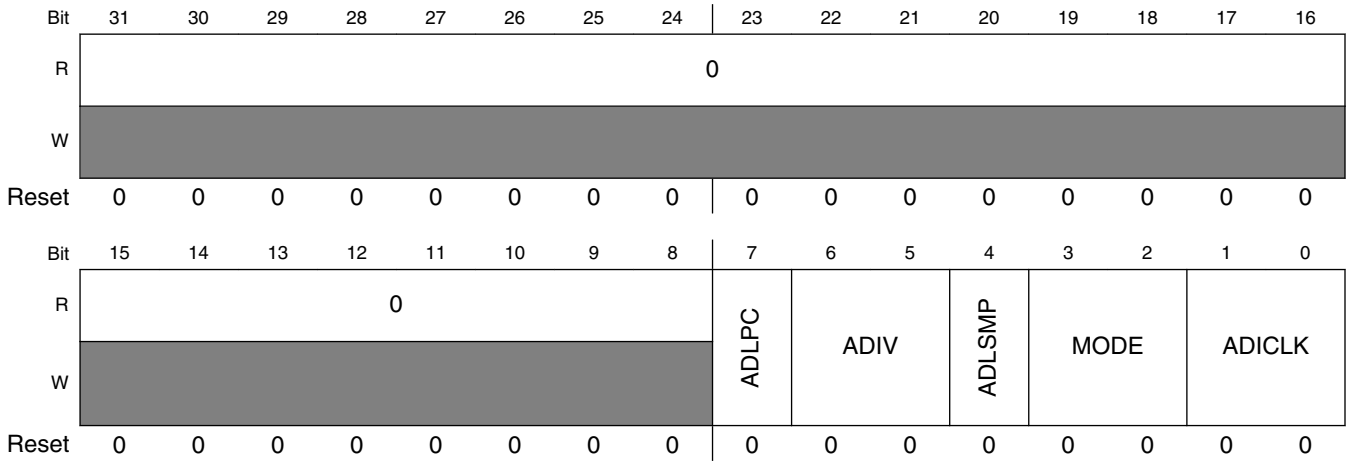
ADC*x*_SC1*n* field descriptions (continued)

Field	Description	
	01000	When DIFF=0, AD8 is selected as input; when DIFF=1, it is reserved.
	01001	When DIFF=0, AD9 is selected as input; when DIFF=1, it is reserved.
	01010	When DIFF=0, AD10 is selected as input; when DIFF=1, it is reserved.
	01011	When DIFF=0, AD11 is selected as input; when DIFF=1, it is reserved.
	01100	When DIFF=0, AD12 is selected as input; when DIFF=1, it is reserved.
	01101	When DIFF=0, AD13 is selected as input; when DIFF=1, it is reserved.
	01110	When DIFF=0, AD14 is selected as input; when DIFF=1, it is reserved.
	01111	When DIFF=0, AD15 is selected as input; when DIFF=1, it is reserved.
	10000	When DIFF=0, AD16 is selected as input; when DIFF=1, it is reserved.
	10001	When DIFF=0, AD17 is selected as input; when DIFF=1, it is reserved.
	10010	When DIFF=0, AD18 is selected as input; when DIFF=1, it is reserved.
	10011	When DIFF=0, AD19 is selected as input; when DIFF=1, it is reserved.
	10100	When DIFF=0, AD20 is selected as input; when DIFF=1, it is reserved.
	10101	When DIFF=0, AD21 is selected as input; when DIFF=1, it is reserved.
	10110	When DIFF=0, AD22 is selected as input; when DIFF=1, it is reserved.
	10111	When DIFF=0, AD23 is selected as input; when DIFF=1, it is reserved.
	11000	Reserved.
	11001	Reserved.
	11010	When DIFF=0, Temp Sensor (single-ended) is selected as input; when DIFF=1, Temp Sensor (differential) is selected as input.
	11011	When DIFF=0,Bandgap (single-ended) is selected as input; when DIFF=1, Bandgap (differential) is selected as input.
	11100	Reserved.
	11101	When DIFF=0,V_{REFSH} is selected as input; when DIFF=1, -V_{REFSH} (differential) is selected as input. Voltage reference selected is determined by SC2[REFSEL].
	11110	When DIFF=0,V_{REFSL} is selected as input; when DIFF=1, it is reserved. Voltage reference selected is determined by SC2[REFSEL].
	11111	Module is disabled.

33.3.2 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 + 8h 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...

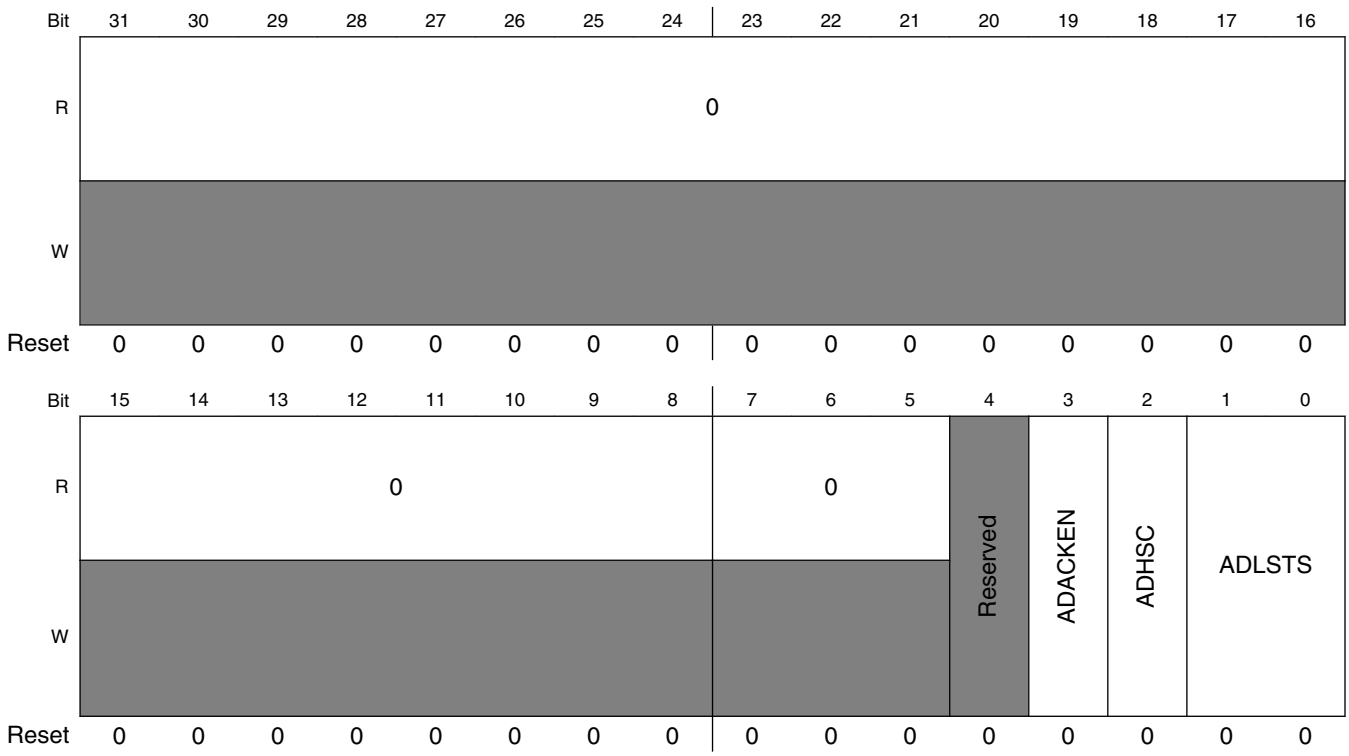
ADCx_CFG1 field descriptions (continued)

Field	Description
	0 Short sample time. 1 Long sample time.
3–2 MODE	Conversion mode selection Selects the ADC resolution mode. 00 When DIFF=0:It is single-ended 8-bit conversion; when DIFF=1, it is differential 9-bit conversion with 2's complement output. 01 When DIFF=0:It is single-ended 12-bit conversion ; when DIFF=1, it is differential 13-bit conversion with 2's complement output. 10 When DIFF=0:It is single-ended 10-bit conversion. ; when DIFF=1, it is differential 11-bit conversion with 2's complement output 11 When DIFF=0:It is single-ended 16-bit conversion..; when DIFF=1, it is differential 16-bit conversion with 2's complement output
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/2) 10 Alternate clock (ALTCLK) 11 Asynchronous clock (ADACK)

33.3.3 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 + Ch 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.

33.3.4 ADC Data Result Register (ADCx_R*n*)

The data result registers (Rn) contain the result of an ADC conversion of the channel selected by the corresponding status and channel control register (SC1A:SC1n). For every status and channel control register, there is a corresponding data result register.

Unused bits in R n are cleared in unsigned right-aligned modes and carry the sign bit (MSB) in sign-extended 2's complement modes. For example, when configured for 10-bit single-ended mode, D[15:10] are cleared. When configured for 11-bit differential mode, D[15:10] carry the sign bit, that is, bit 10 extended through bit 15.

The following table describes the behavior of the data result registers in the different modes of operation.

Table 33-2. 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 differential	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Signed 2's complement
16-bit single-ended	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Unsigned right justified
13-bit differential	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	Sign-extended 2's complement

Table continues on the next page...

Table 33-2. Data result register description (continued)

Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format
12-bit single-ended	0	0	0	0	D	D	D	D	D	D	D	D	D	D	D	D	Unsigned right-justified
11-bit differential	S	S	S	S	S	S	D	D	D	D	D	D	D	D	D	D	Sign-extended 2's complement
10-bit single-ended	0	0	0	0	0	0	D	D	D	D	D	D	D	D	D	D	Unsigned right-justified
9-bit differential	S	S	S	S	S	S	S	S	D	D	D	D	D	D	D	D	Sign-extended 2's complement
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 + 10h offset + (4d × i), where i=0d to 1d

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																D															
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	0	0	0

ADC*x*_R*n* field descriptions

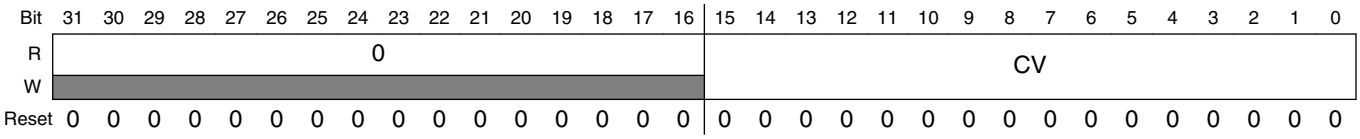
Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
D	Data result

33.3.5 Compare Value Registers (ADC*x*_CV*n*)

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 + 18h offset + (4d × i), where i=0d to 1d



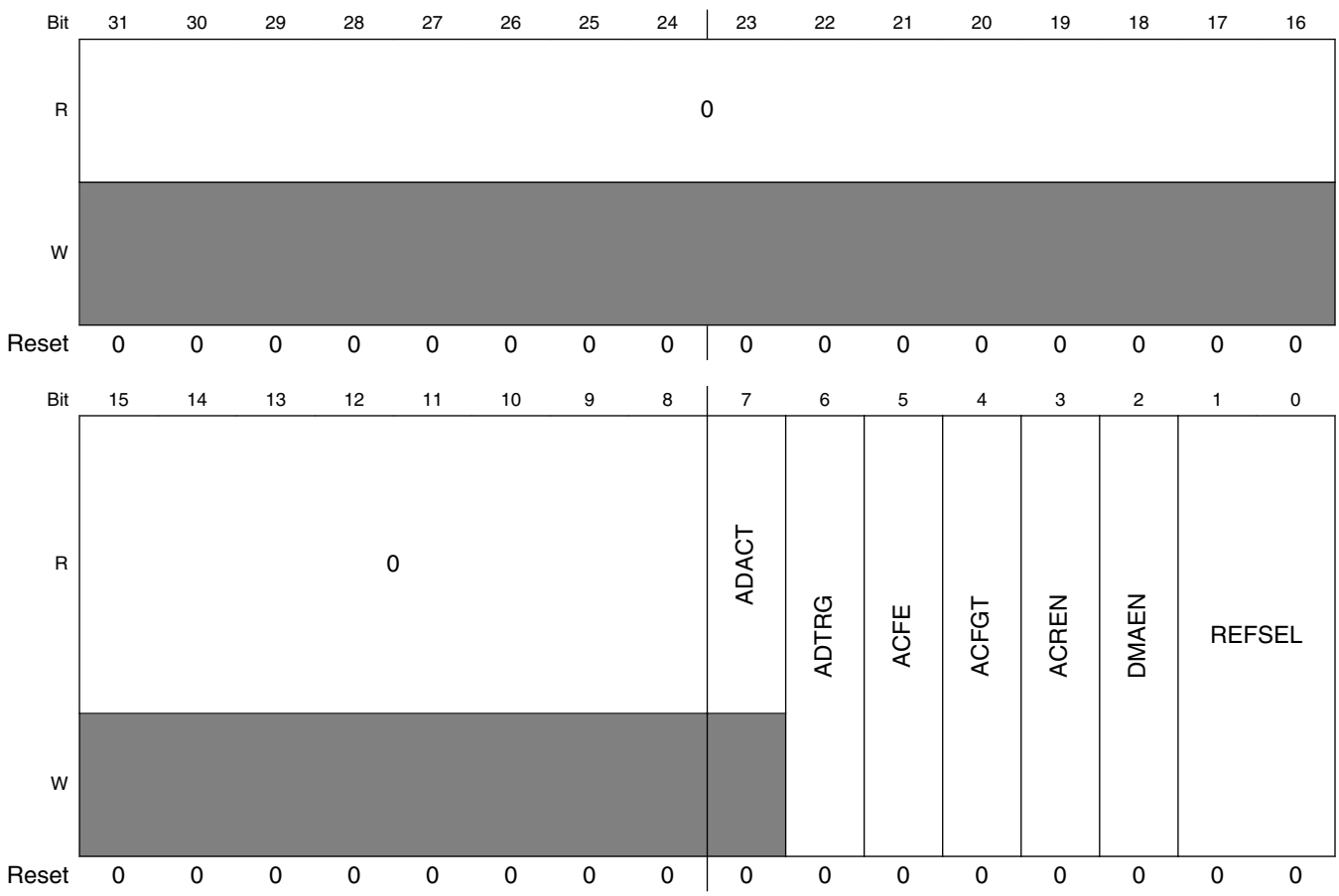
ADCx_CV*n* field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CV	Compare Value.

33.3.6 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 + 20h 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: • 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}

Table continues on the next page...

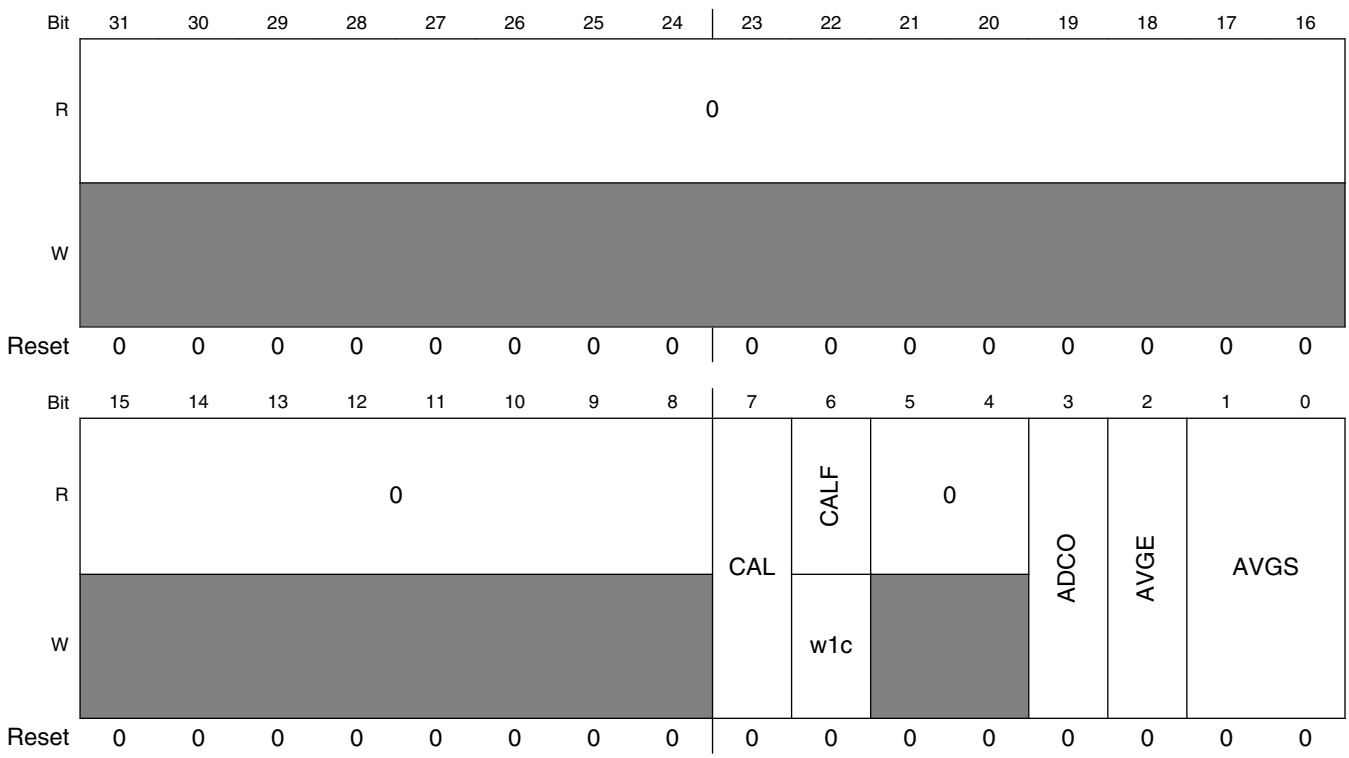
ADCx_SC2 field descriptions (continued)

Field	Description
	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 Reserved 11 Reserved

33.3.7 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 + 24h 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)

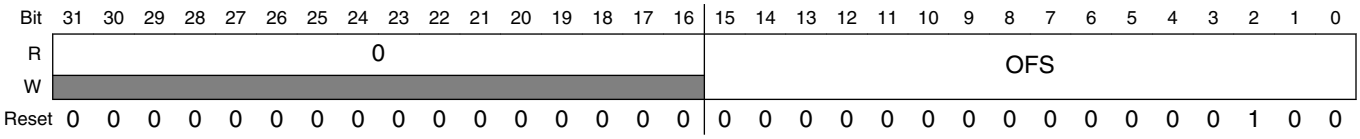
Field	Description
6 CALF	Calibration Failed Flag Displays the result of the calibration sequence. The calibration sequence will fail if SC2[ADTRG] = 1, any ADC register is written, or any stop mode is entered before the calibration sequence completes. Writing 1 to CALF clears it. 0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.
5–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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.

33.3.8 ADC Offset Correction Register (ADCx_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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: Base address + 28h offset



ADCx_OFS field descriptions

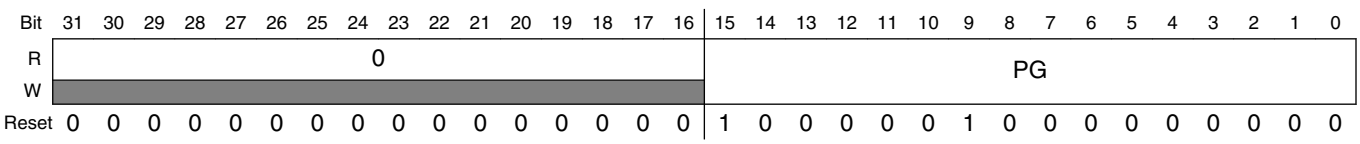
Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
OFS	Offset Error Correction Value

33.3.9 ADC Plus-Side Gain Register (ADCx_PG)

The Plus-Side Gain Register (PG) contains the gain error correction for the plus-side input in differential mode or 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 PG[15] and PG[14]. 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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: Base address + 2Ch offset



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

33.3.10 ADC Minus-Side Gain Register (ADCx_MG)

The Minus-Side Gain Register (MG) contains the gain error correction for the minus-side input in differential mode. This register is ignored in single-ended mode. MG, a 16-bit real number in binary format, is the gain adjustment factor, with the radix point fixed between MG[15] and MG[14]. 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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: Base address + 30h 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																MG															
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_MG field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MG	Minus-Side Gain

33.3.11 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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: Base address + 34h 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

33.3.12 ADC Plus-Side General Calibration Value Register (ADCx_CLPS)

For more information, see CLPD register description.

Address: Base address + 38h 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

33.3.13 ADC Plus-Side General Calibration Value Register (ADCx_CLP4)

For more information, see CLPD register description.

Address: Base address + 3Ch 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.

Table continues on the next page...

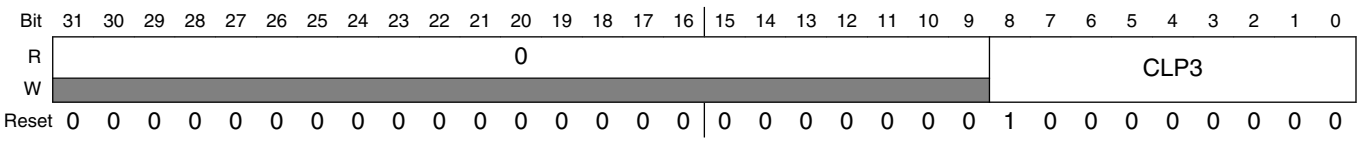
ADCx_CLP4 field descriptions (continued)

Field	Description
CLP4	Calibration Value Calibration Value

33.3.14 ADC Plus-Side General Calibration Value Register (ADCx_CLP3)

For more information, see CLPD register description.

Address: Base address + 40h offset



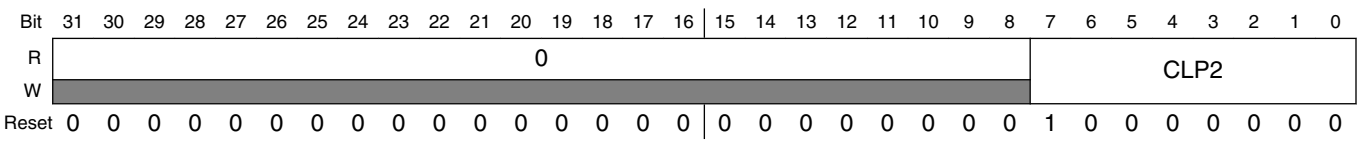
ADCx_CLP3 field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP3	Calibration Value Calibration Value

33.3.15 ADC Plus-Side General Calibration Value Register (ADCx_CLP2)

For more information, see CLPD register description.

Address: Base address + 44h offset



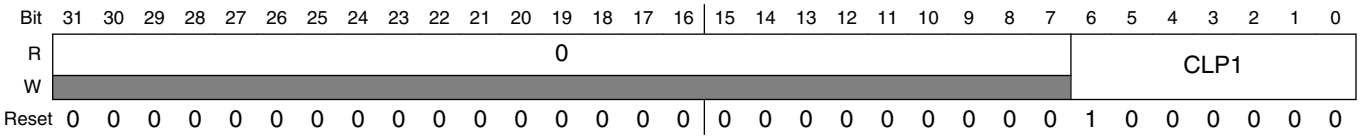
ADCx_CLP2 field descriptions

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

33.3.16 ADC Plus-Side General Calibration Value Register (ADCx_CLP1)

For more information, see CLPD register description.

Address: Base address + 48h offset



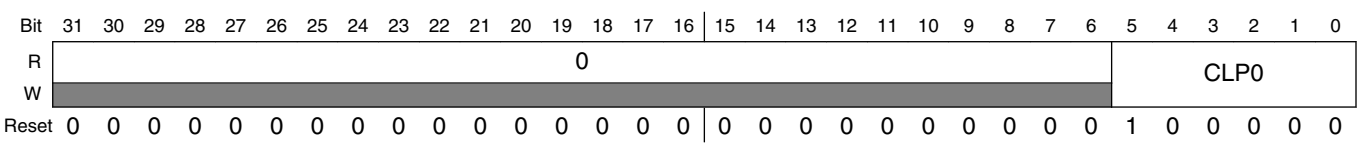
ADCx_CLP1 field descriptions

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

33.3.17 ADC Plus-Side General Calibration Value Register (ADCx_CLP0)

For more information, see CLPD register description.

Address: Base address + 4Ch offset



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

33.3.18 ADC Minus-Side General Calibration Value Register (ADCx_CLMD)

The Minus-Side General Calibration Value (CLMx) registers contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLM0[5:0], CLM1[6:0], CLM2[7:0], CLM3[8:0], CLM4[9:0], CLMS[5:0], and CLMD[5:0]. CLMx 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.

For more information regarding the calibration procedure, please refer to the Calibration function section.

Address: Base address + 54h 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																										CLMD					
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_CLMD field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLMD	Calibration Value Calibration Value

33.3.19 ADC Minus-Side General Calibration Value Register (ADCx_CLMS)

For more information, see CLMD register description.

Address: Base address + 58h 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																										CLMS					
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_CLMS field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADCx_CLMS field descriptions (continued)

Field	Description
CLMS	Calibration Value Calibration Value

33.3.20 ADC Minus-Side General Calibration Value Register (ADCx_CLM4)

For more information, see CLMD register description.

Address: Base address + 5Ch 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																CLM4															
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_CLM4 field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM4	Calibration Value Calibration Value

33.3.21 ADC Minus-Side General Calibration Value Register (ADCx_CLM3)

For more information, see CLMD register description.

Address: Base address + 60h 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																							CLM3								
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_CLM3 field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM3	Calibration Value Calibration Value

33.3.22 ADC Minus-Side General Calibration Value Register (ADCx_CLM2)

For more information, see CLMD register description.

Address: Base address + 64h 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																CLM2															
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	1	0	0	0	0	0	0	0

ADCx_CLM2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM2	Calibration Value Calibration Value

33.3.23 ADC Minus-Side General Calibration Value Register (ADCx_CLM1)

For more information, see CLMD register description.

Address: Base address + 68h 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																									CLM1						
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	1	0	0	0	0	0	0

ADCx_CLM1 field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM1	Calibration Value Calibration Value

33.3.24 ADC Minus-Side General Calibration Value Register (ADCx_CLM0)

For more information, see CLMD register description.

Address: Base address + 6Ch 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																CLM0															
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_CLM0 field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM0	Calibration Value Calibration Value

33.4 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.

33.4.1 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.

33.4.2 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.

33.4.3 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-specific ADC information 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.

33.4.4 Conversion control

Conversions can be performed as determined by CFG1[MODE] and SC1n[DIFF] 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

33.4.4.1 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.

33.4.4.2 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.

33.4.4.3 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.

33.4.4.4 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}.

33.4.4.5 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] and SC1n[DIFF]
- 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 33-3. 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 33-4. 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 33-5. Base conversion time (BCT)

Mode	Base conversion time (BCT)
8b single-ended	17 ADCK cycles
9b differential	27 ADCK cycles
10b single-ended	20 ADCK cycles
11b differential	30 ADCK cycles
12b single-ended	20 ADCK cycles
13b differential	30 ADCK cycles

Table continues on the next page...

Table 33-5. Base conversion time (BCT) (continued)

Mode	Base conversion time (BCT)
16b single-ended	25 ADCK cycles
16b differential	34 ADCK cycles

Table 33-6. 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 33-7. 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.

33.4.4.6 Conversion time examples

The following examples use the Equation 1 on page 675, and the information provided in Table 33-3 through Table 33-7.

33.4.4.6.1 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 675, and the information provided in Table 33-3 through Table 33-7. The table below lists the variables of Equation 1 on page 675.

Table 33-8. Typical conversion time

Variable	Time
SFCAdder	5 ADCK cycles + 5 bus clock cycles
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.

33.4.4.6.2 Long conversion time configuration

A configuration for long ADC conversion is:

- 16-bit differential mode with the bus clock selected as the input clock source
- 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 675, and the information provided in Table 33-3 through Table 33-7. The following table lists the variables of the Equation 1 on page 675.

Table 33-9. 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.

33.4.4.6.3 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 675, and the information provided in Table 33-3 through Table 33-7. The table below lists the variables of Equation 1 on page 675.

Table 33-10. 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.

33.4.4.7 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.

33.4.5 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 33-11. 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.

33.4.6 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, the minus-side calibration values, and the plus-side calibration values. The offset calibration value is automatically stored in the ADC offset correction register (OFS), and the plus-side and minus-side calibration values are automatically stored in the ADC plus-side and minus-side calibration registers, CLPx and CLMx. The user must configure the ADC correctly prior to calibration, and must generate the plus-side and minus-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 differential/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.
7. Repeat the procedure for the minus-side gain calibration value.

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 and minus-side gain, and plus-side and minus-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.

33.4.7 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. The same bits are used in 9-bit differential mode because OFS[15] indicates the sign bit, which maps to D[8]. For 16-bit differential mode, OFS[15:0] are directly subtracted from the conversion result data D[15:0]. 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; for a differential conversion it is 0x8000.

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.

33.4.8 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).

33.4.9 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, bus clock divided by two; and 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.

33.4.10 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.

33.4.10.1 Normal Stop mode with ADACK disabled

If the asynchronous clock, ADACK, is not 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.

33.4.10.2 Normal Stop mode with ADACK enabled

If ADACK is selected as the conversion clock, the ADC continues operation during Normal Stop mode. See the chip-specific ADC information for configuration information for this device.

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.

33.4.11 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.

33.5 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 or 16-bit, 13-bit, 11-bit, or 9-bit differential resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. For information used in this example, refer to Table 33-6, Table 33-7, and Table 33-8.

Note

Hexadecimal values are designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

33.5.1 ADC module initialization example

33.5.1.1 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 select whether conversions will be single-ended or differential and to enable or disable conversion complete interrupts. Also, select the input channel which can be used to perform conversions.

33.5.1.2 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, differential 11-
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 5 DIFF 0 Single-ended conversion selected.
        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.
```

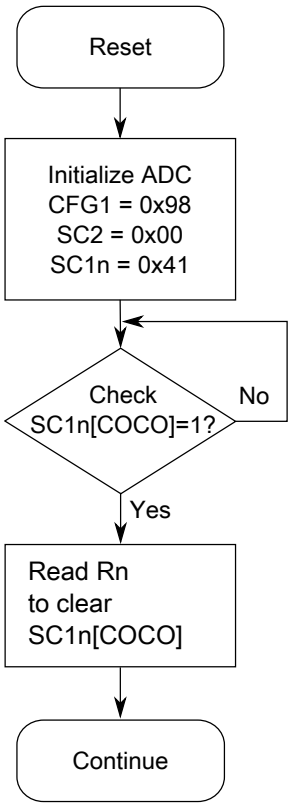


Figure 33-2. Initialization flowchart example

33.6 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.

33.6.1 External pins and routing

33.6.1.1 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.

33.6.1.2 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.

33.6.1.3 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.

33.6.2 Sources of error

33.6.2.1 Sampling error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy.

$$RAS + RADIN = SC / (FMAX * NUMTAU * CADIN)$$

Figure 33-3. 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.

33.6.2.2 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.

33.6.2.3 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.

33.6.2.4 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.

33.6.2.5 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.

33.6.2.6 Code jitter, non-monotonicity, and missing codes

Analog-to-digital converters are susceptible to three special forms of error:

- Code jitter: Code jitter occurs when 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 occurs 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 34
Comparator (CMP)

34.1 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.

34.1.1 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
- DMA transfer support
  - A comparison event can be selected to trigger a DMA transfer
- Functional in all modes of operation
- The window and filter functions are not available in the following modes:
  - Stop
  - VLPS

34.1.2 6-bit DAC key features

The 6-bit DAC has the following 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

34.1.3 ANMUX key features

The ANMUX has the following features:

- Two 8-to-1 channel mux
- Operational over the entire supply range

34.1.4 CMP, DAC and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.

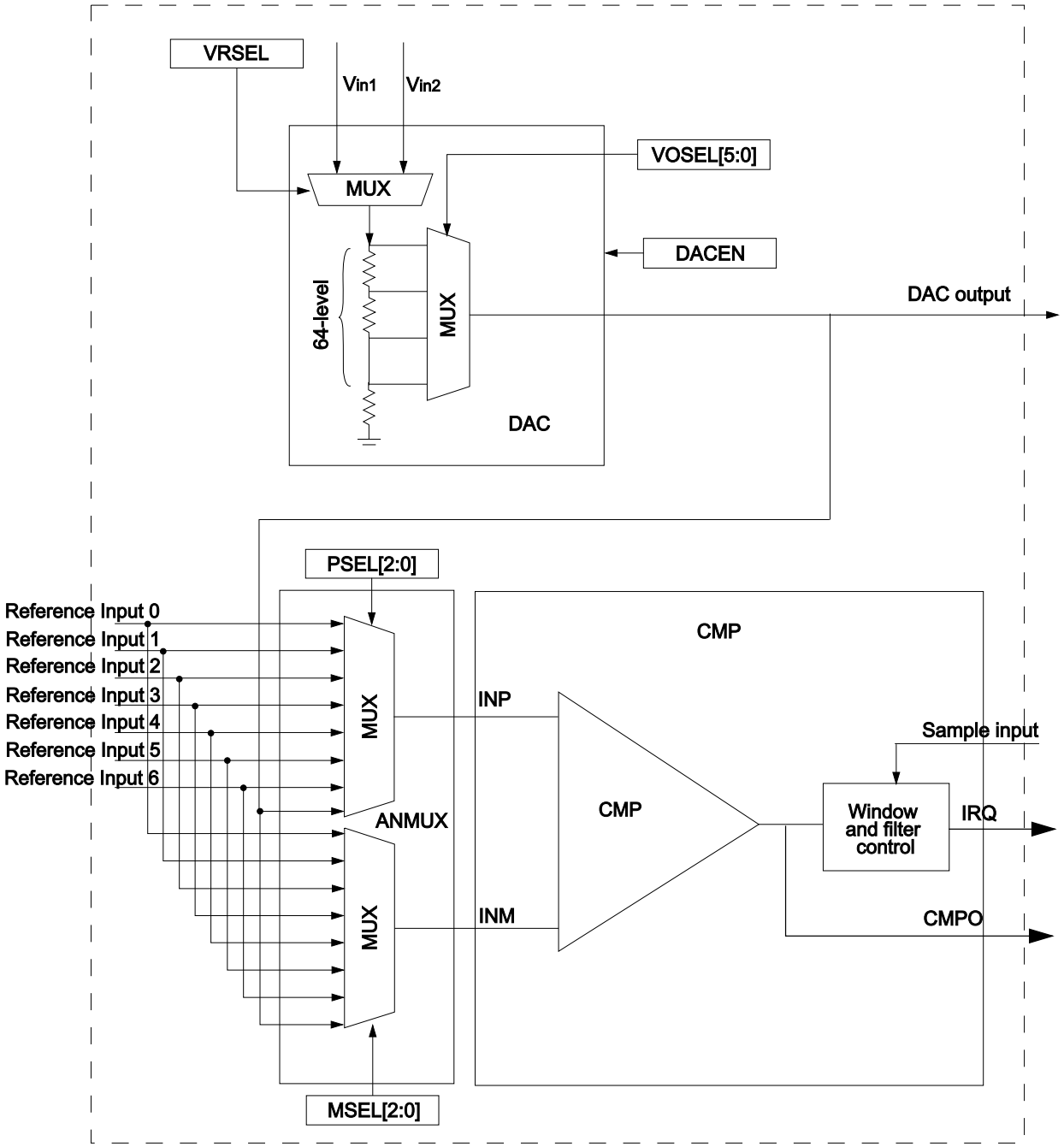


Figure 34-1. CMP, DAC and ANMUX block diagram

34.1.5 CMP block diagram

The following figure shows the block diagram for the CMP module.

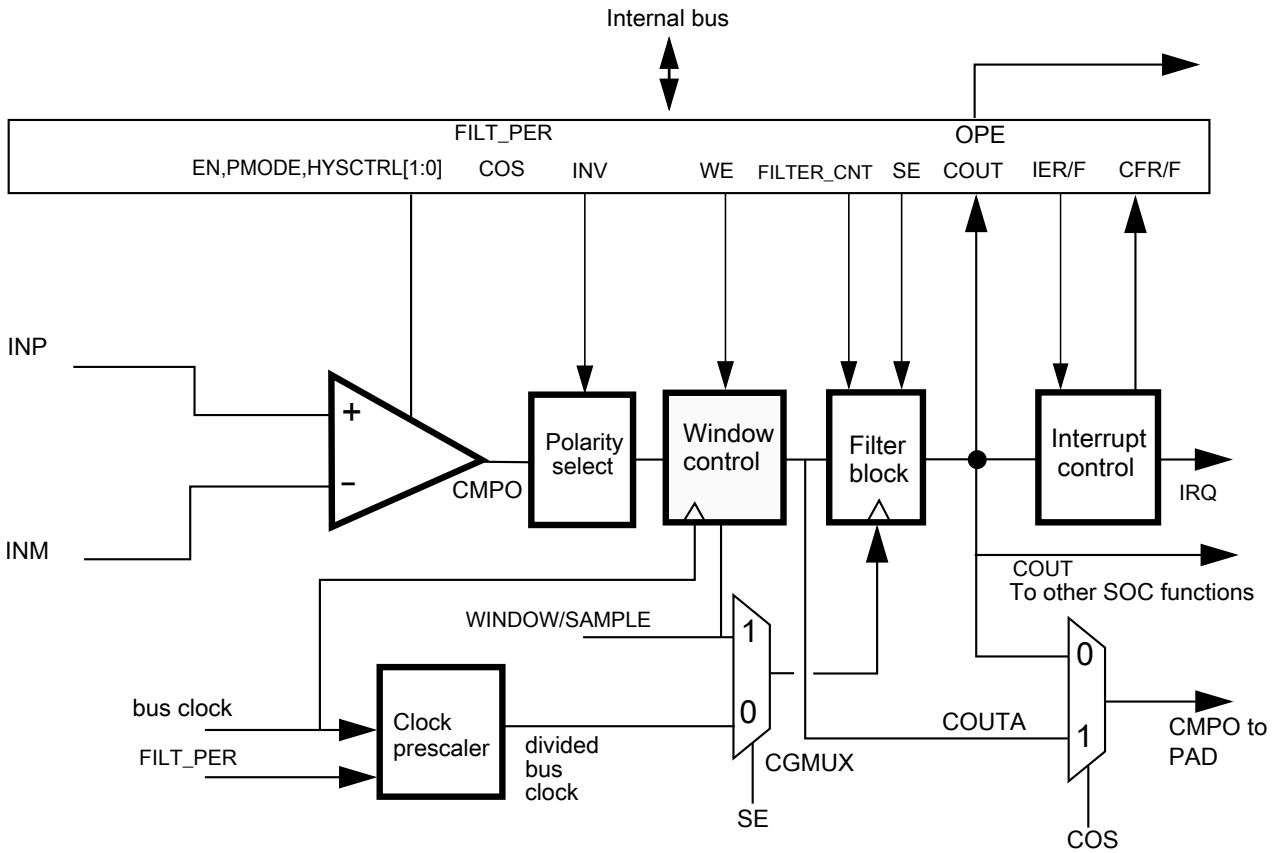


Figure 34-2. Comparator module block diagram

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.

34.2 Memory map/register definitions

CMP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_3000	CMP Control Register 0 (CMP0_CR0)	8	R/W	00h	34.2.1/700
4007_3001	CMP Control Register 1 (CMP0_CR1)	8	R/W	00h	34.2.2/701
4007_3002	CMP Filter Period Register (CMP0_FPR)	8	R/W	00h	34.2.3/703
4007_3003	CMP Status and Control Register (CMP0_SCR)	8	R/W	00h	34.2.4/703
4007_3004	DAC Control Register (CMP0_DACCR)	8	R/W	00h	34.2.5/704
4007_3005	MUX Control Register (CMP0_MUXCR)	8	R/W	00h	34.2.6/705
4007_3008	CMP Control Register 0 (CMP1_CR0)	8	R/W	00h	34.2.1/700
4007_3009	CMP Control Register 1 (CMP1_CR1)	8	R/W	00h	34.2.2/701
4007_300A	CMP Filter Period Register (CMP1_FPR)	8	R/W	00h	34.2.3/703
4007_300B	CMP Status and Control Register (CMP1_SCR)	8	R/W	00h	34.2.4/703
4007_300C	DAC Control Register (CMP1_DACCR)	8	R/W	00h	34.2.5/704
4007_300D	MUX Control Register (CMP1_MUXCR)	8	R/W	00h	34.2.6/705

34.2.1 CMP Control Register 0 (CMPx_CR0)

Address: Base address + 0h offset

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

CMPx_CR0 field descriptions

Field	Description
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

Table continues on the next page...

CMPx_CR0 field descriptions (continued)

Field	Description
	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 10 Level 2 11 Level 3

34.2.2 CMP Control Register 1 (CMPx_CR1)

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read Write	SE	WE	TRIGM	PMODE	INV	COS	OPE	EN
Reset	0	0	0	0	0	0	0	0

CMPx_CR1 field descriptions

Field	Description
7 SE	Sample Enable At any given time, either SE or WE can be set. It is mandatory request to not set SE and WE both at a given time. 0 Sampling mode is not selected. 1 Sampling mode is selected.
6 WE	Windowing Enable

Table continues on the next page...

CMPx_CR1 field descriptions (continued)

Field	Description
	At any given time, either SE or WE can be set. It is mandatory request to not set SE and WE both at a given time. 0 Windowing mode is not selected. 1 Windowing mode is selected.
5 TRIGM	Trigger Mode Enable CMP and DAC are configured to CMP Trigger mode when CMP_CR1[TRIGM] is set to 1. In addition, the CMP should be enabled. If the DAC is to be used as a reference to the CMP, it should also be enabled. CMP Trigger mode depends on an external timer resource to periodically enable the CMP and 6-bit DAC in order to generate a triggered compare. Upon setting TRIGM, the CMP and DAC are placed in a standby state until an external timer resource trigger is received. See the chip configuration for details about the external timer resource. 0 Trigger mode is disabled. 1 Trigger mode is enabled.
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.
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.

CMPx_CR1 field descriptions (continued)

Field	Description

34.2.3 CMP Filter Period Register (CMPx_FPR)

Address: Base address + 2h offset

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

CMPx_FPR field descriptions

Field	Description
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.

34.2.4 CMP Status and Control Register (CMPx_SCR)

Address: Base address + 3h offset

Bit	7	6	5	4	3	2	1	0
Read	0	DMAEN	0	IER	IEF	CFR	CFF	COUT
Write						w1c	w1c	
Reset	0	0	0	0	0	0	0	0

CMPx_SCR field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 DMAEN	DMA Enable Control Enables the DMA transfer triggered from the CMP module. When this field is set, a DMA request is asserted when CFR or CFF is set. 0 DMA is disabled. 1 DMA is enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CMPx_SCR field descriptions (continued)

Field	Description
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 is edge sensitive . 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 is edge sensitive . 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.

34.2.5 DAC Control Register (CMPx_DACCR)

Address: Base address + 4h offset

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
7 DACEN	DAC Enable Enables the DAC. When the DAC is disabled, it is powered down to conserve power.

Table continues on the next page...

CMPx_DACCR field descriptions (continued)

Field	Description
	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} .

34.2.6 MUX Control Register (CMPx_MUXCR)

Address: Base address + 5h offset

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

CMPx_MUXCR field descriptions

Field	Description
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.

Table continues on the next page...

CMPx_MUXCR field descriptions (continued)

Field	Description
	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

34.3 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].

34.3.1 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 34-1. 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 , 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.

34.3.1.1 Disabled mode (# 1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

34.3.1.2 Continuous mode (#s 2A & 2B)

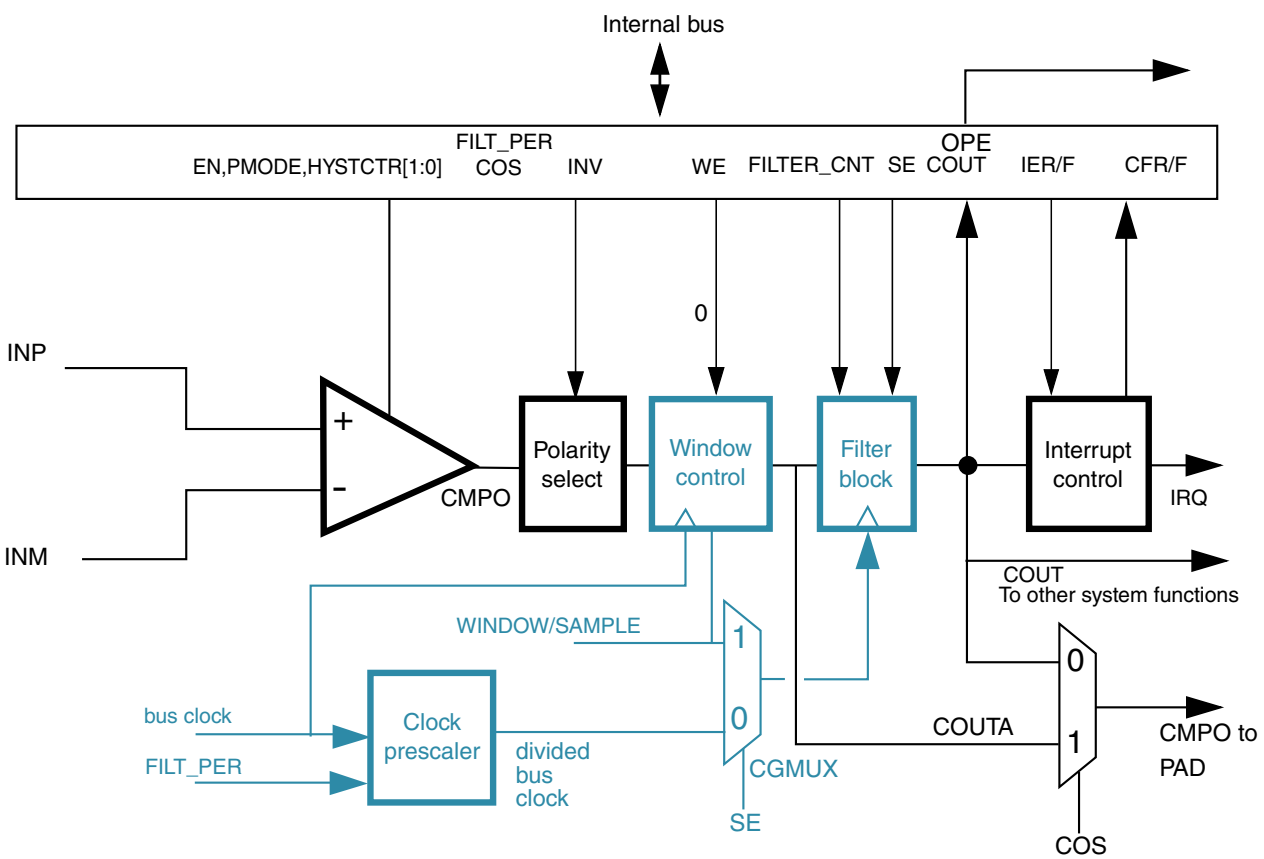


Figure 34-3. Comparator operation in Continuous mode

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.

34.3.1.3 Sampled, Non-Filtered mode (#s 3A & 3B)

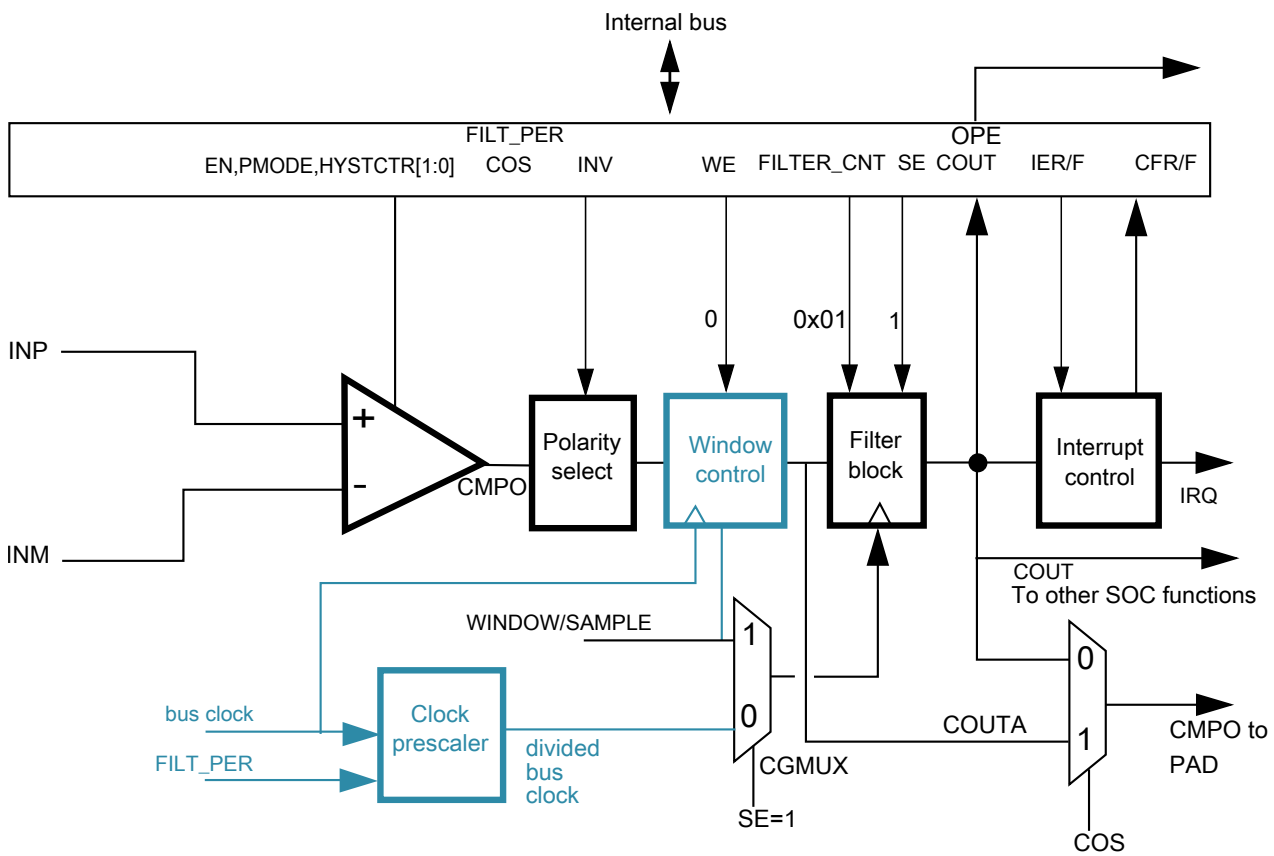


Figure 34-4. Sampled, Non-Filtered (# 3A): sampling point externally driven

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).

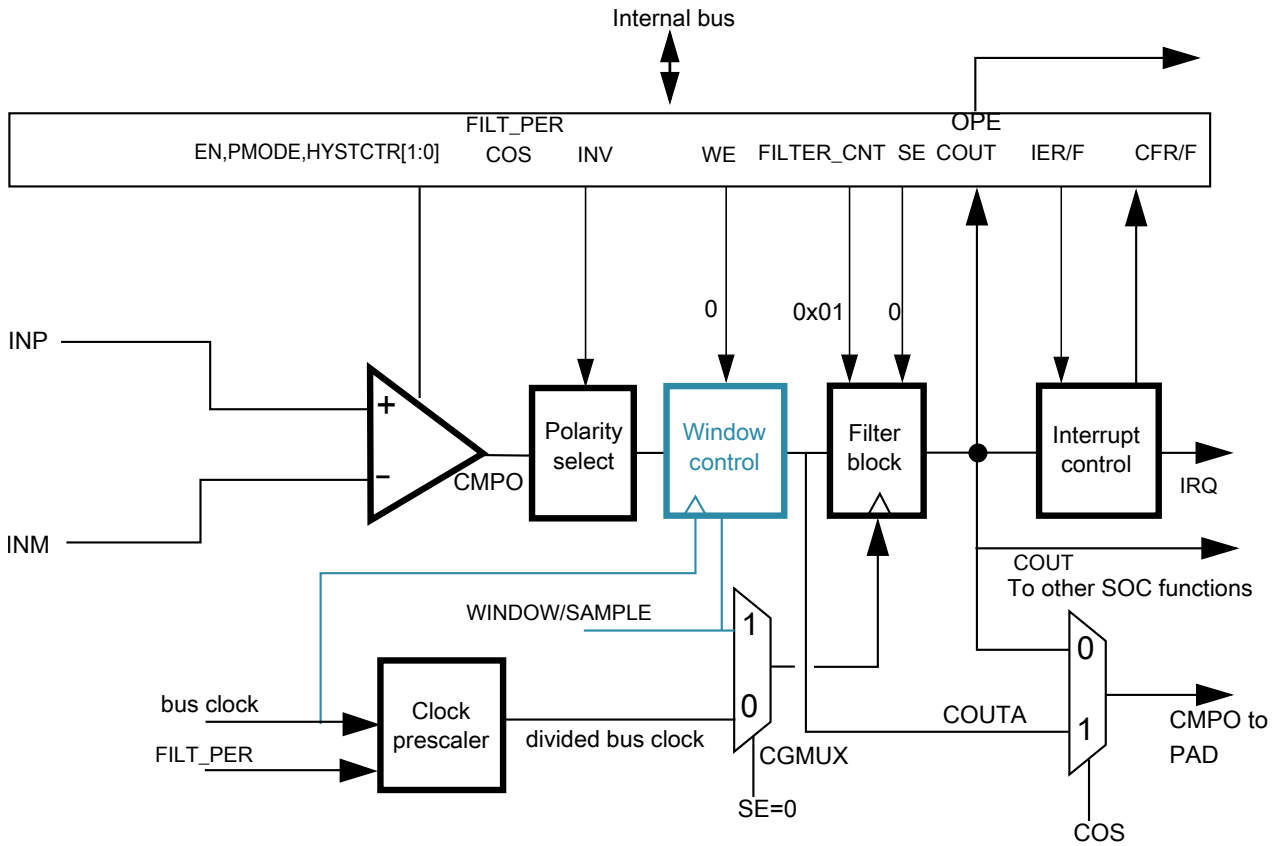


Figure 34-5. Sampled, Non-Filtered (# 3B): sampling interval internally derived

34.3.1.4 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.

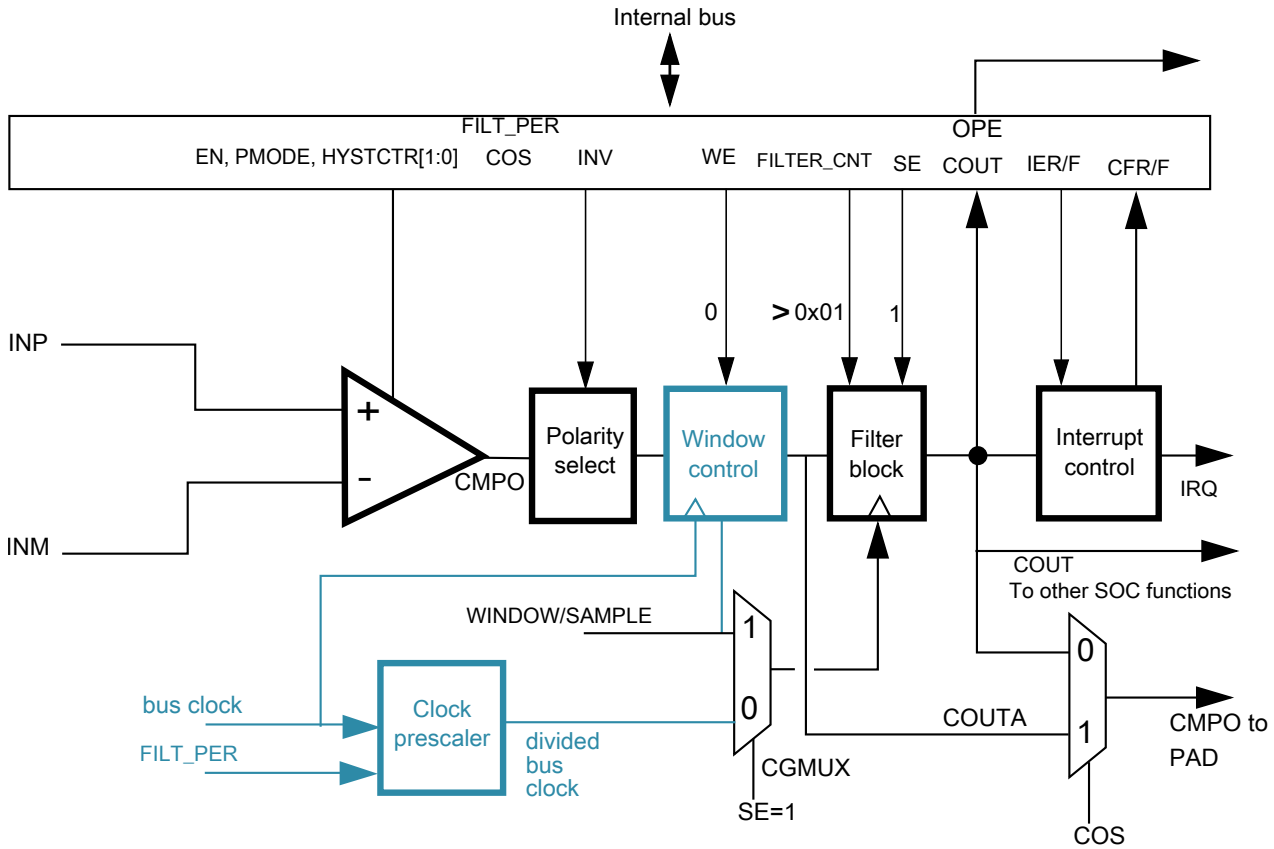


Figure 34-6. Sampled, Filtered (# 4A): sampling point externally driven

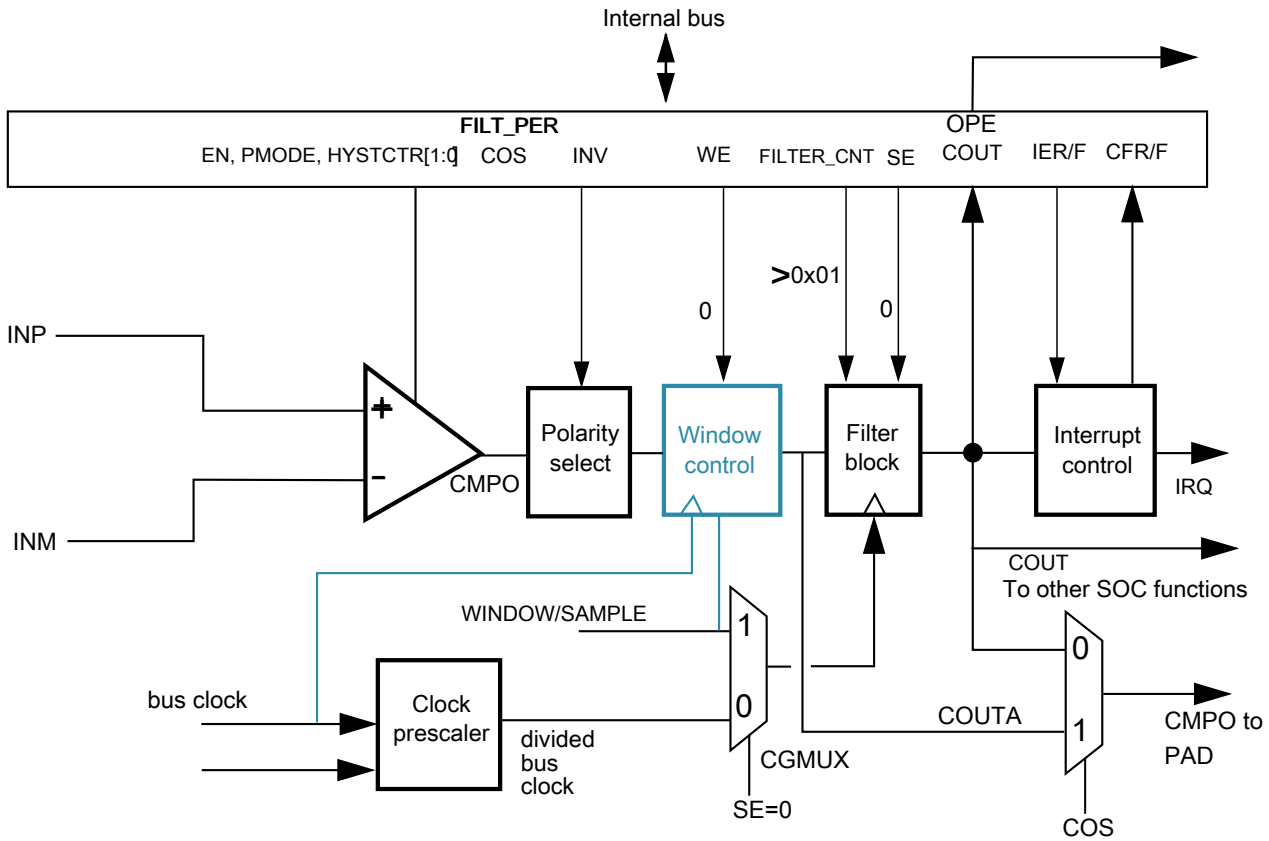


Figure 34-7. Sampled, Filtered (# 4B): sampling point internally derived

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.

34.3.1.5 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.

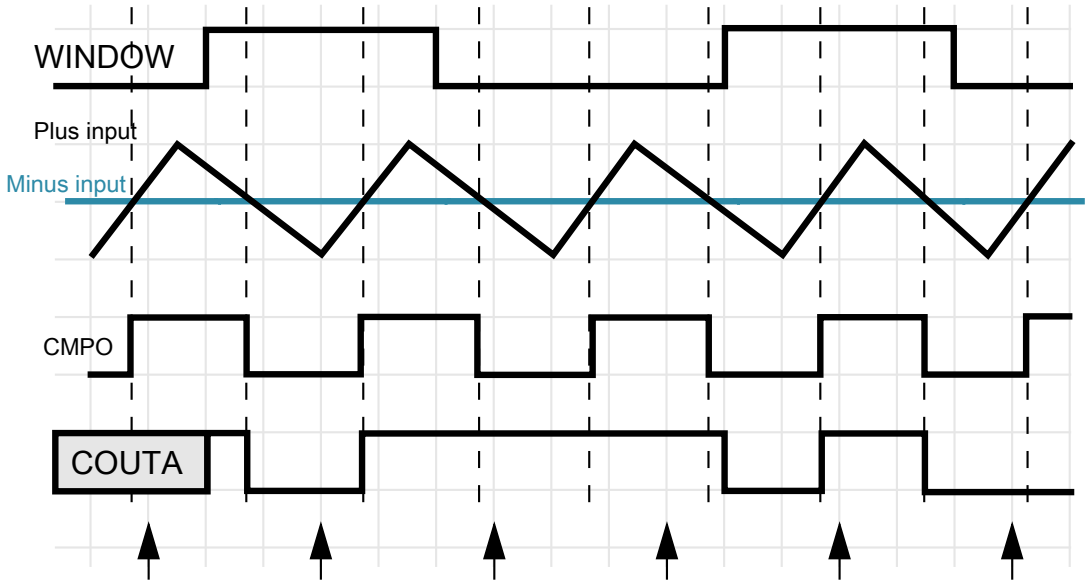


Figure 34-8. Windowed mode operation

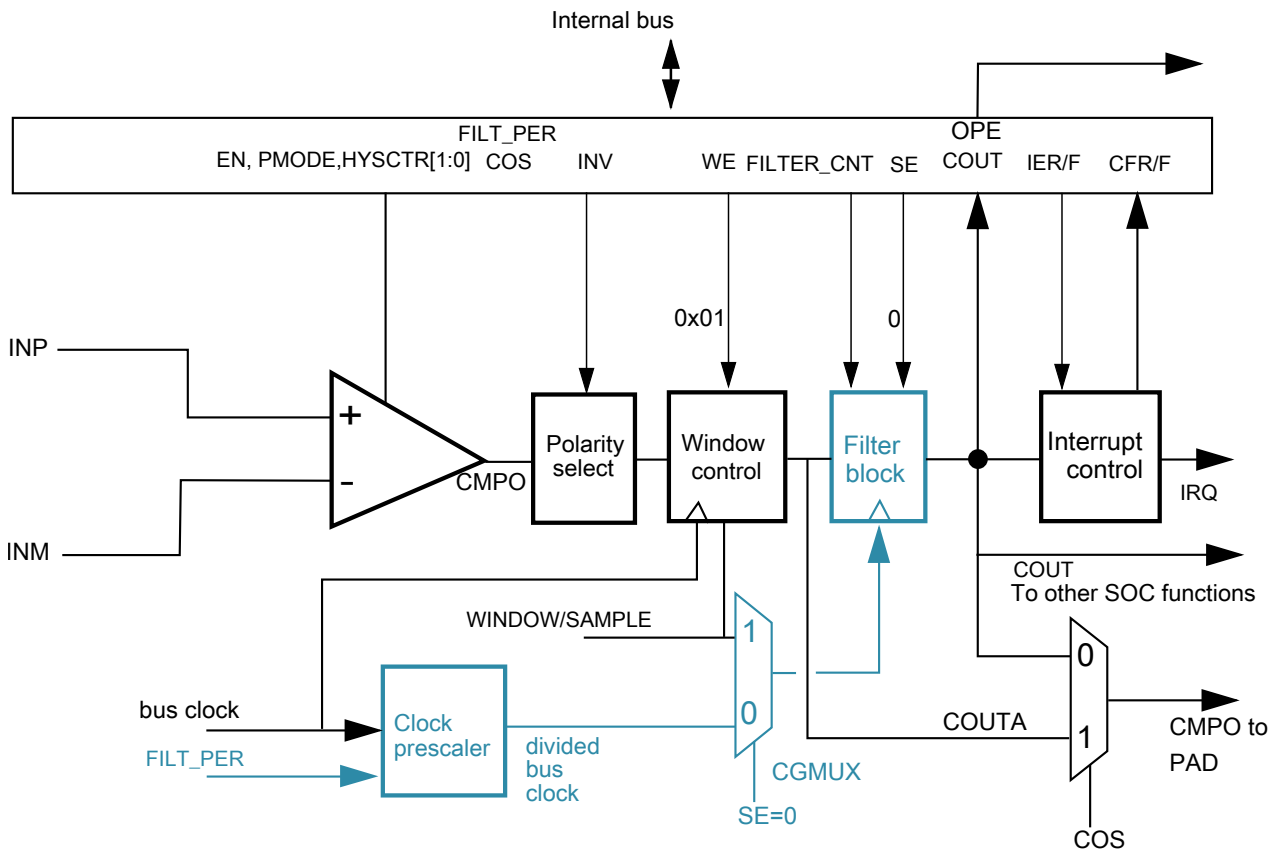


Figure 34-9. Windowed mode

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.

34.3.1.6 Windowed/Resampled mode (# 6)

The following figure uses the same input stimulus shown in Figure 34-8, 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.

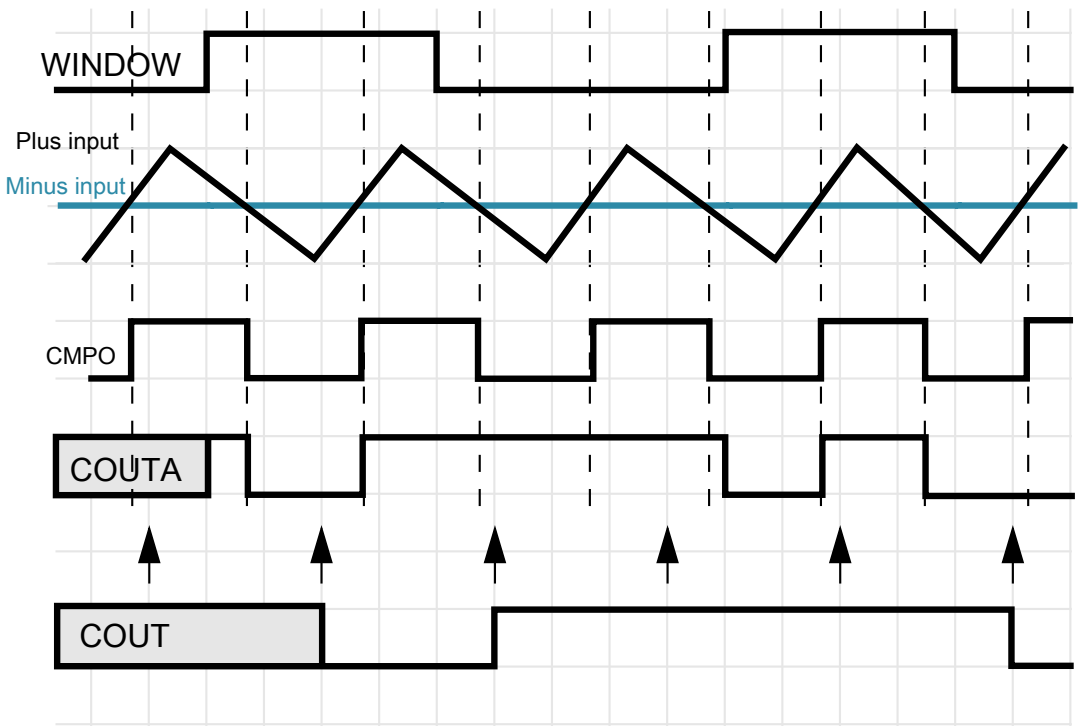


Figure 34-10. Windowed/resampled mode operation

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.

34.3.1.7 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.

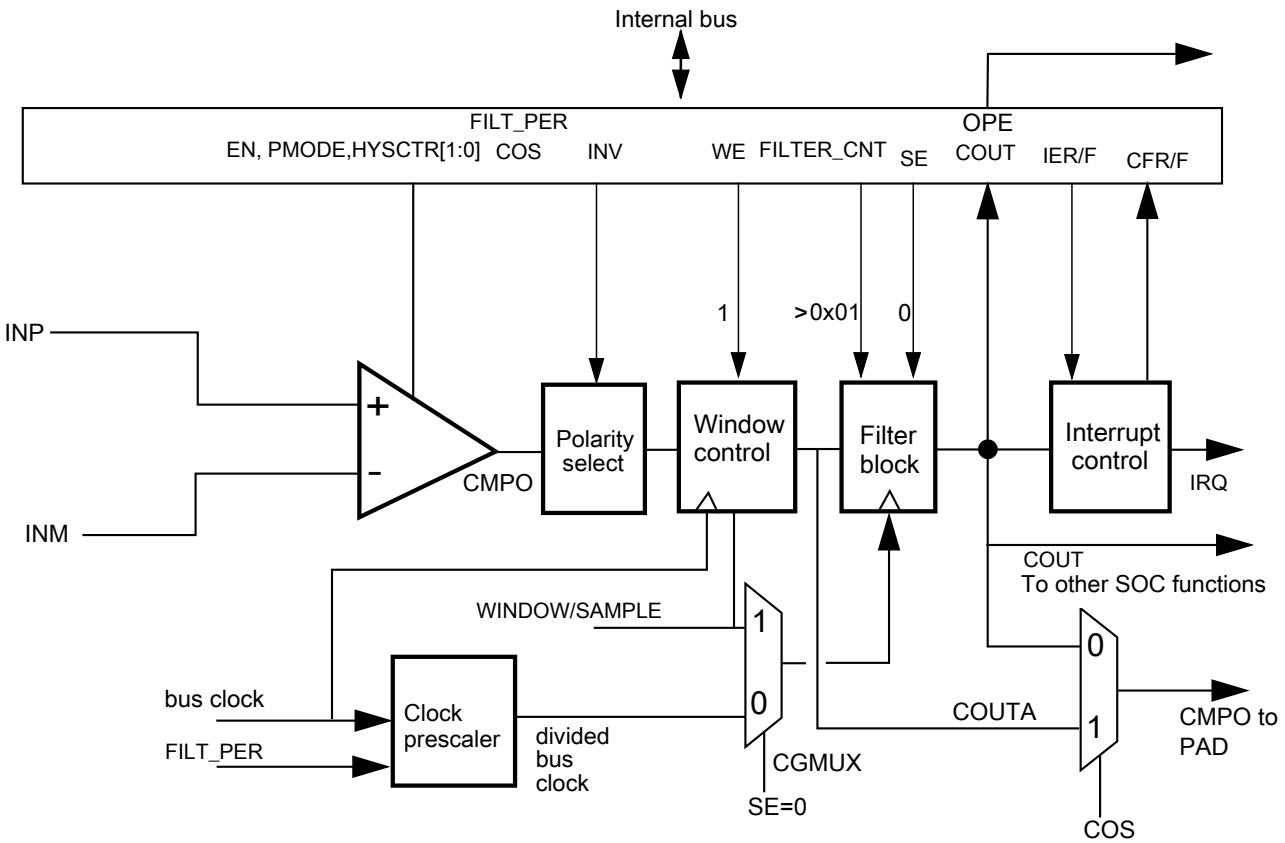


Figure 34-11. Windowed/Filtered mode

34.3.2 Power modes

34.3.2.1 Wait mode operation

During Wait and VLPW modes, the CMP, if enabled, continues to operate normally and a CMP interrupt can wake the MCU.

34.3.2.2 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.

34.3.2.3 Background Debug Mode Operation

When the microcontroller is in active background debug mode, the CMP continues to operate normally.

34.3.3 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.

34.3.4 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.

34.3.4.1 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.

34.3.4.2 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 34-2. 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.

34.4 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

34.5 DMA support

Normally, the CMP generates a CPU interrupt if there is a change on the COUT. When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt instead. When the DMA has completed the transfer, it sends a transfer completing indicator that deasserts the DMA transfer request and clears the flag to allow a subsequent change on comparator output to occur and force another DMA request.

The comparator can remain functional in STOP modes.

When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], or both, the corresponding change on COUT forces a DMA transfer request to wake up the system from STOP modes. After the data transfer has finished, system will go back to STOP modes. Refer to DMA chapters in the device reference manual for the asynchronous DMA function for details.

34.6 CMP Asynchronous DMA support

The comparator can remain functional in STOP modes.

When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], or both, the corresponding change on COUT forces a DMA transfer request to wake up the system from STOP modes. After the data transfer has finished, system will go back to STOP modes. Refer to DMA chapters in the device reference manual for the asynchronous DMA function for details.

34.7 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.

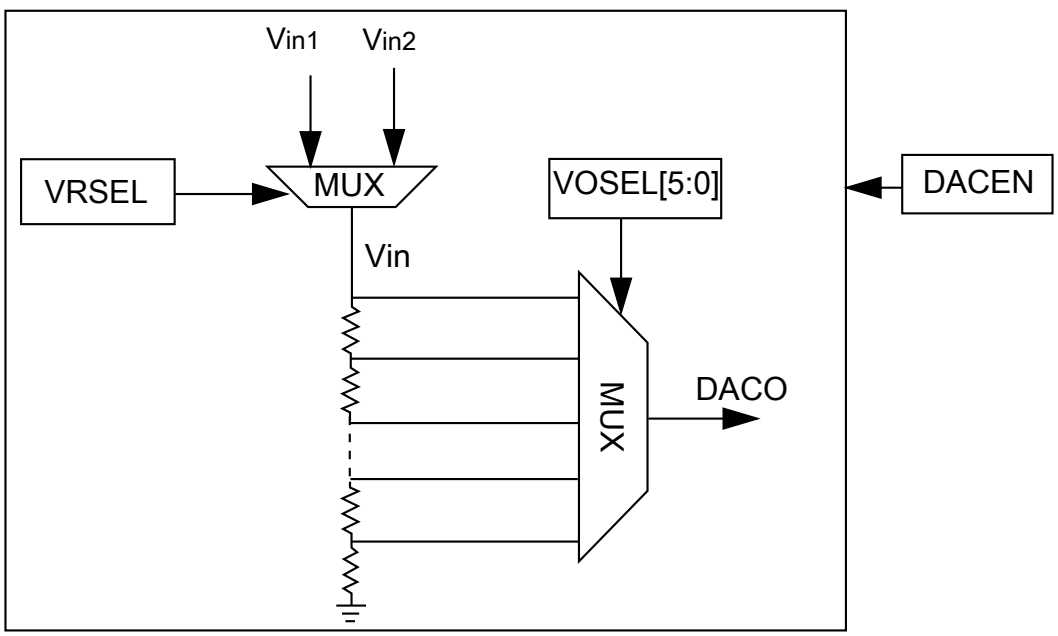


Figure 34-12. 6-bit DAC block diagram

34.8 DAC functional description

This section provides DAC functional description information.

34.8.1 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

34.9 DAC resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

34.10 DAC clocks

This module has a single clock input, the bus clock.

34.11 DAC interrupts

This module has no interrupts.

34.12 CMP Trigger Mode

CMP and DAC are configured to CMP Trigger mode when CMP_CR1[TRIGM] is set to 1.

In addition, the CMP must be enabled. If the DAC is to be used as a reference to the CMP, it must also be enabled.

CMP Trigger mode depends on an external timer resource to periodically enable the CMP and 6-bit DAC in order to generate a triggered compare.

Upon setting TRIGM, the CMP and DAC are placed in a standby state until an external timer resource trigger is received.

See the chip configuration chapter for details about the external timer resource.

Chapter 35
12-bit Digital-to-Analog Converter (DAC)

35.1 Introduction

NOTE

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

The 12-bit digital-to-analog converter (DAC) is a low-power, general-purpose DAC. The output of the DAC can be placed on an external pin or set as one of the inputs to the analog comparator, op-amps, or ADC.

35.2 Features

The features of the DAC module include:

- On-chip programmable reference generator output. The voltage output range is from 1⁄4096 V_{in} to V_{in}, and the step is 1⁄4096 V_{in}, where V_{in} is the input voltage.
- V_{in} can be selected from two reference sources
- Static operation in Normal Stop mode
- 2-word data buffer supported with configurable watermark and multiple operation modes
- DMA support

35.3 Block diagram

The block diagram of the DAC module is as follows:

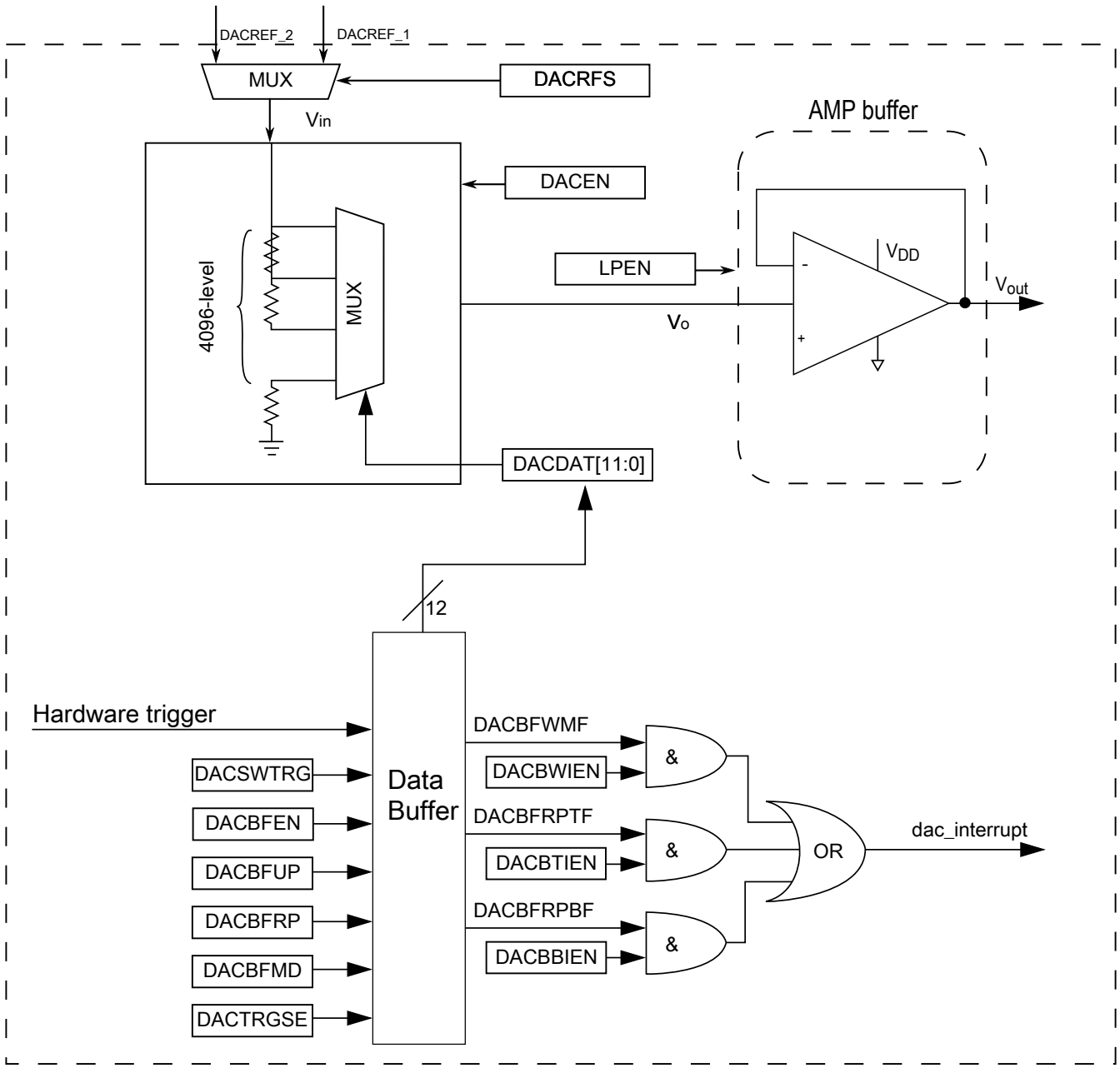


Figure 35-1. DAC block diagram

35.4 Memory map/register definition

The DAC has registers to control analog comparator and programmable voltage divider to perform the digital-to-analog functions.

DAC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_F000	DAC Data Low Register (DAC0_DAT0L)	8	R/W	00h	35.4.1/725
4003_F001	DAC Data High Register (DAC0_DAT0H)	8	R/W	00h	35.4.2/725
4003_F002	DAC Data Low Register (DAC0_DAT1L)	8	R/W	00h	35.4.1/725
4003_F003	DAC Data High Register (DAC0_DAT1H)	8	R/W	00h	35.4.2/725
4003_F020	DAC Status Register (DAC0_SR)	8	R/W	06h	35.4.3/726
4003_F021	DAC Control Register (DAC0_C0)	8	R/W	00h	35.4.4/727
4003_F022	DAC Control Register 1 (DAC0_C1)	8	R/W	00h	35.4.5/728
4003_F023	DAC Control Register 2 (DAC0_C2)	8	R/W	01h	35.4.6/729

35.4.1 DAC Data Low Register (DAC*x*_DAT*n*L)

Address: 4003_F000h base + 0h offset + (2d × i), where i=0d to 1d

Bit	7	6	5	4	3	2	1	0
Read Write	DATA0							
Reset	0	0	0	0	0	0	0	0

DAC*x*_DAT*n*L field descriptions

Field	Description
DATA0	DATA0 When the DAC buffer is not enabled, DATA[11:0] controls the output voltage based on the following formula: $V_{out} = V_{in} * (1 + DACDAT0[11:0])/4096$ When the DAC buffer is enabled, DATA is mapped to the 16-word buffer.

35.4.2 DAC Data High Register (DAC*x*_DAT*n*H)

Address: 4003_F000h base + 1h offset + (2d × i), where i=0d to 1d

Bit	7	6	5	4	3	2	1	0
Read	0				DATA1			
Write								
Reset	0	0	0	0	0	0	0	0

DAC*x*_DAT*n*H field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DATA1	DATA1

Table continues on the next page...

DACx_DATnH field descriptions (continued)

Field	Description
	When the DAC Buffer is not enabled, DATA[11:0] controls the output voltage based on the following formula. $V_{out} = V_{in} * (1 + DACDAT0[11:0])/4096$ When the DAC buffer is enabled, DATA[11:0] is mapped to the 16-word buffer.

35.4.3 DAC Status Register (DACx_SR)

If DMA is enabled, the flags can be cleared automatically by DMA when the DMA request is done. Writing 0 to a field clears it whereas writing 1 has no effect. After reset, DACBFRPTF is set and can be cleared by software, if needed. The flags are set only when the data buffer status is changed.

NOTE

Do not use 32/16-bit accesses to this register.

NOTE

A watermark interrupt is generated if C0[DACBWIEN] is set after any write to DAC data register, since there is only 1 word depth for the buffer watermark. It is recommended not to set C0[DACBWIEN] to avoid repeated entry into the watermark interrupt.

Address: 4003_F000h base + 20h offset = 4003_F020h

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

DACx_SR field descriptions

Field	Description
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 DACBFWMF	DAC Buffer Watermark Flag 0 The DAC buffer read pointer has not reached the watermark level. 1 The DAC buffer read pointer has reached the watermark level.
1 DACBFRPTF	DAC Buffer Read Pointer Top Position Flag 0 The DAC buffer read pointer is not zero. 1 The DAC buffer read pointer is zero.
0 DACBFRPBF	DAC Buffer Read Pointer Bottom Position Flag 0 The DAC buffer read pointer is not equal to C2[DACBFUP]. 1 The DAC buffer read pointer is equal to C2[DACBFUP].

35.4.4 DAC Control Register (DACx_C0)

NOTE

Do not use 32- or 16-bit accesses to this register.

NOTE

A watermark interrupt is generated if C0[DACBWIEN] is set after any write to DAC data register, since there is only 1 word depth for the buffer watermark. It is recommended not to set C0[DACBWIEN] to avoid repeated entry into the watermark interrupt.

Address: 4003_F000h base + 21h offset = 4003_F021h

Bit	7	6	5	4	3	2	1	0
Read	DACEN	DACRFS	DACTRGSEL	0	LPEN	DACBWIEN	DACBTIEN	DACBBIEN
Write				DACSWTRG				
Reset	0	0	0	0	0	0	0	0

DACx_C0 field descriptions

Field	Description
7 DACEN	DAC Enable Starts the Programmable Reference Generator operation. 0 The DAC system is disabled. 1 The DAC system is enabled.
6 DACRFS	DAC Reference Select 0 The DAC selects DACREF_1 as the reference voltage. 1 The DAC selects DACREF_2 as the reference voltage.
5 DACTRGSEL	DAC Trigger Select 0 The DAC hardware trigger is selected. 1 The DAC software trigger is selected.
4 DACSWTRG	DAC Software Trigger Active high. This is a write-only field, which always reads 0. If DAC software trigger is selected and buffer is enabled, writing 1 to this field will advance the buffer read pointer once. 0 The DAC soft trigger is not valid. 1 The DAC soft trigger is valid.
3 LPEN	DAC Low Power Control **NOTE:** See the 12-bit DAC electrical characteristics of the device data sheet for details on the impact of the modes below.

Table continues on the next page...

DACx_C0 field descriptions (continued)

Field	Description
	0 High-Power mode 1 Low-Power mode
2 DACBWIEN	DAC Buffer Watermark Interrupt Enable 0 The DAC buffer watermark interrupt is disabled. 1 The DAC buffer watermark interrupt is enabled.
1 DACBTIEN	DAC Buffer Read Pointer Top Flag Interrupt Enable 0 The DAC buffer read pointer top flag interrupt is disabled. 1 The DAC buffer read pointer top flag interrupt is enabled.
0 DACBBIEN	DAC Buffer Read Pointer Bottom Flag Interrupt Enable 0 The DAC buffer read pointer bottom flag interrupt is disabled. 1 The DAC buffer read pointer bottom flag interrupt is enabled.

35.4.5 DAC Control Register 1 (DACx_C1)

NOTE

Do not use 32- or 16-bit accesses to this register.

Address: 4003_F000h base + 22h offset = 4003_F022h

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

DACx_C1 field descriptions

Field	Description
7 DMAEN	DMA Enable Select 0 DMA is disabled. 1 DMA is enabled. When DMA is enabled, the DMA request will be generated by original interrupts. The interrupts will not be presented on this module at the same time.
6–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 DACBFMD	DAC Buffer Work Mode Select 0 Normal mode 1 One-Time Scan mode
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

DACx_C1 field descriptions (continued)

Field	Description
0 DACBFEN	DAC Buffer Enable 0 Buffer read pointer is disabled. The converted data is always the first word of the buffer. 1 Buffer read pointer is enabled. The converted data is the word that the read pointer points to. It means converted data can be from any word of the buffer.

35.4.6 DAC Control Register 2 (DACx_C2)

Address: 4003_F000h base + 23h offset = 4003_F023h

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

DACx_C2 field descriptions

Field	Description
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 DACBFRP	DAC Buffer Read Pointer Keeps the current value of the buffer read pointer.
3–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 DACBFUP	DAC Buffer Upper Limit Selects the upper limit of the DAC buffer. The buffer read pointer cannot exceed it.

35.5 Functional description

The 12-bit DAC module can select one of the two reference inputs—DACREF_1 and DACREF_2 as the DAC reference voltage, V_{in} by C0 [DACRFS]. See the chip-specific DAC information to determine the source options for DACREF_1 and DACREF_2.

When the DAC is enabled, it converts the data in DACDAT0[11:0] or the data from the DAC data buffer to a stepped analog output voltage. The output voltage range is from V_{in} to $V_{in}/4096$, and the step is $V_{in}/4096$.

35.5.1 DAC data buffer operation

When the DAC is enabled and the buffer is not enabled, the DAC module always converts the data in DAT0 to the analog output voltage.

When both the DAC and the buffer are enabled, the DAC converts the data in the data buffer to analog output voltage. The data buffer read pointer advances to the next word whenever a hardware or software trigger event occurs.

The data buffer can be configured to operate in Normal mode, One-Time Scan mode. When the buffer operation is switched from one mode to another, the read pointer does not change. The read pointer can be set to any value between 0 and C2[DACBFUP] by writing C2[DACBFRP].

35.5.1.1 DAC data buffer interrupts

There are several interrupts and associated flags that can be configured for the DAC buffer. SR[DACBFRPBF] is set when the DAC buffer read pointer reaches the DAC buffer upper limit, that is, C2[DACBFRP] = C2[DACBFUP]. SR[DACBFRPTF] is set when the DAC read pointer is equal to the start position, 0. Finally, SR[DACBFWMF] is set when the DAC buffer read pointer has reached the position defined by C1[DACBFWM]. C1[DACBFWM] can be used to generate an interrupt when the DAC buffer read pointer is between 1 to 4 words from C2[DACBFUP].

35.5.1.2 Modes of DAC data buffer operation

The following table describes the different modes of data buffer operation for the DAC module.

Table 35-1. Modes of DAC data buffer operation

Modes	Description
Buffer Normal mode	This is the default mode. The buffer works as a circular buffer. The read pointer increases by one, every time the trigger occurs. When the read pointer reaches the upper limit, it goes to 0 directly in the next trigger event.
Buffer One-time Scan mode	The read pointer increases by 1 every time the trigger occurs. When it reaches the upper limit, it stops there. If read pointer is reset to the address other than the upper limit, it will increase to the upper address and stop there again. **NOTE:** If the software set the read pointer to the upper limit, the read pointer will not advance in this mode.

35.5.2 DMA operation

When DMA is enabled, DMA requests are generated instead of interrupt requests. The DMA Done signal clears the DMA request.

The status register flags are still set and are cleared automatically when the DMA completes.

35.5.3 Resets

During reset, the DAC is configured in the default mode and is disabled.

35.5.4 Low-Power mode operation

The following table shows the wait mode and the stop mode operation of the DAC module.

Table 35-2. Modes of operation

Modes of operation	Description
Wait mode	The DAC will operate normally, if enabled.
Stop mode	If enabled, the DAC module continues to operate in Normal Stop mode and the output voltage will hold the value before stop. In low-power stop modes, the DAC is fully shut down.

NOTE

The assignment of module modes to core modes is chip-specific. For module-to-core mode assignments, see the chapter that describes how modules are configured.

Chapter 36
FlexTimer Module (FTM)

36.1 Introduction

NOTE

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

The FlexTimer module (FTM) is a two-to-eight channel timer that supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications. The FTM time reference is a 16-bit counter that can be used as an unsigned or signed counter.

NOTE

The number of channels supported can vary for each instance of the FTM module on a chip. See the chip-specific FTM information to see how many channels are supported for each module instance. For example, if a module instance supports only six channels, references to channel numbers 6 and 7 do not apply for that instance.

36.1.1 FlexTimer philosophy

The FlexTimer is built upon a simple timer, the Timer PWM Module – TPM, used for many years on our HCS08 family of 8-bit microcontrollers. The FlexTimer extends the functionality to meet the demands of motor control, digital lighting solutions, and power conversion, while providing low cost and backwards compatibility with the TPM module.

Several key enhancements are made:

- Signed up counter
- Deadtime insertion hardware
- Fault control inputs

- Enhanced triggering functionality
- Initialization and polarity control

All of the features common with the TPM have fully backwards compatible register assignments. The FlexTimer can also use code on the same core platform without change to perform the same functions.

Motor control and power conversion features have been added through a dedicated set of registers and defaults turn off all new features. The new features, such as hardware deadtime insertion, polarity, fault control, and output forcing and masking, greatly reduce loading on the execution software and are usually each controlled by a group of registers.

FlexTimer input triggers can be from comparators, ADC, or other submodules to initiate timer functions automatically. These triggers can be linked in a variety of ways during integration of the sub modules so please note the options available for used FlexTimer configuration.

More than one FlexTimers may be synchronized to provide a larger timer with their counters incrementing in unison, assuming the initialization, the input clocks, the initial and final counting values are the same in each FlexTimer.

All main user access registers are buffered to ease the load on the executing software. A number of trigger options exist to determine which registers are updated with this user defined data.

36.1.2 Features

The FTM features include:

- FTM source clock is selectable.
    - The source clock can be the system clock, the fixed frequency clock, or an external clock
    - Fixed frequency clock is an additional clock input to allow the selection of an on chip clock source other than the system clock
    - Selecting external clock connects FTM clock to a chip level input pin therefore allowing to synchronize the FTM counter with an off chip clock source
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit counter
    - It can be a free-running counter or a counter with initial and final value
    - The counting can be up or up-down

- Each channel can be configured for input capture, output compare, or edge-aligned PWM mode
- In Input Capture mode:
  - The capture can occur on rising edges, falling edges or both edges
  - An input filter can be selected for some channels
- In Output Compare mode the output signal can be set, cleared, or toggled on match
- All channels can be configured for center-aligned PWM mode
- Each pair of channels can be combined to generate a PWM signal with independent control of both edges of PWM signal
- The FTM channels can operate as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs
- The deadtime insertion is available for each complementary pair
- Generation of match triggers
- Initialization trigger
- Software control of PWM outputs
- Up to 4 fault inputs for global fault control
- The polarity of each channel is configurable
- The generation of an interrupt per channel
- The generation of an interrupt when the counter overflows
- The generation of an interrupt when the fault condition is detected
- Synchronized loading of write buffered FTM registers
- Write protection for critical registers
- Backwards compatible with TPM
- Testing of input captures for a stuck at zero and one conditions
- Dual edge capture for pulse and period width measurement
- Quadrature decoder with input filters, relative position counting, and interrupt on position count or capture of position count on external event

36.1.3 Modes of operation

When the chip is in an active BDM mode, the FTM temporarily suspends all counting until the chip returns to normal user operating mode. During Stop mode, all FTM input clocks are stopped, so the FTM is effectively disabled until clocks resume. During Wait mode, the FTM continues to operate normally. If the FTM does not need to produce a real time reference or provide the interrupt sources needed to wake the chip from Wait mode, the power can then be saved by disabling FTM functions before entering Wait mode.

36.1.4 Block diagram

The FTM uses one input/output (I/O) pin per channel, CHn (FTM channel (n)) where n is the channel number (0–7).

The following figure shows the FTM structure. The central component of the FTM is the 16-bit counter with programmable initial and final values and its counting can be up or up-down.

NOTE

The number of channels supported can vary for each instance of the FTM module on a chip. See the chip-specific FTM information to see how many channels are supported for each module instance. For example, if a module instance supports only six channels, references to channel numbers 6 and 7 do not apply for that instance.

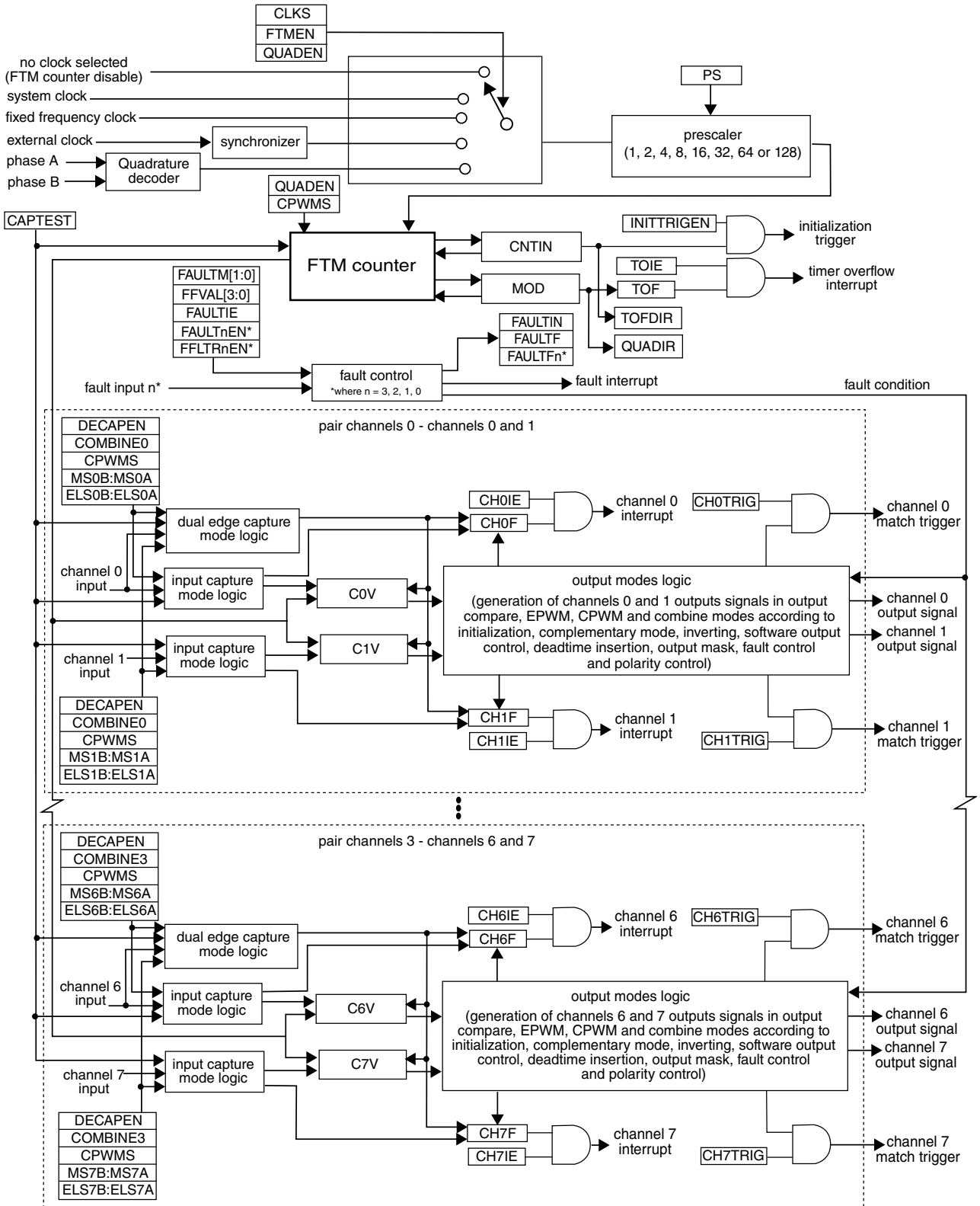


Figure 36-1. FTM block diagram

36.2 FTM signal descriptions

Table 36-1 shows the user-accessible signals for the FTM.

Table 36-1. FTM signal descriptions

Signal	Description	I/O	Function
EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I	The external clock input signal is used as the FTM counter clock if selected by CLKS[1:0] bits in the SC register. This clock signal must not exceed 1/4 of system clock frequency. The FTM counter prescaler selection and settings are also used when an external clock is selected.
CHn	FTM channel (n), where n can be 7-0	I/O	Each FTM channel can be configured to operate either as input or output. The direction associated with each channel, input or output, is selected according to the mode assigned for that channel.
FAULTj	Fault input (j), where j can be 3-0	I	The fault input signals are used to control the CHn channel output state. If a fault is detected, the FAULTj signal is asserted and the channel output is put in a safe state. The behavior of the fault logic is defined by the FAULTM[1:0] control bits in the MODE register and FAULTEN bit in the COMBINEm register. Note that each FAULTj input may affect all channels selectively since FAULTM[1:0] and FAULTEN control bits are defined for each pair of channels. Because there are several FAULTj inputs, maximum of 4 for the FTM module, each one of these inputs is activated by the FAULTjEN bit in the FLTCTRL register.
PHA	Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A.	I	The quadrature decoder phase A input is used as the Quadrature Decoder mode is selected. The phase A input signal is one of the signals that control the FTM counter increment or decrement in the Quadrature Decoder mode.
PHB	Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B.	I	The quadrature decoder phase B input is used as the Quadrature Decoder mode is selected. The phase B input signal is one of the signals that control the FTM counter increment or decrement in the Quadrature Decoder mode.

36.3 Memory map and register definition

36.3.1 Memory map

This section presents a high-level summary of the FTM registers and how they are mapped.

The registers and bits of an unavailable function in the FTM remain in the memory map and in the reset value, but they have no active function.

Note

Do not write in the region from the CNTIN register through the PWMLOAD register when FTMEN = 0.

NOTE

The number of channels supported can vary for each instance of the FTM module on a chip. See the chip-specific FTM information to see how many channels are supported for each module instance. For example, if a module instance supports only six channels, references to channel numbers 6 and 7 do not apply for that instance.

36.3.2 Register descriptions

Accesses to reserved addresses result in transfer errors. Registers for absent channels are considered reserved.

FTM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_6000	Status And Control (FTM3_SC)	32	R/W	0000_0000h	36.3.3/748
4002_6004	Counter (FTM3_CNT)	32	R/W	0000_0000h	36.3.4/749
4002_6008	Modulo (FTM3_MOD)	32	R/W	0000_0000h	36.3.5/750
4002_600C	Channel (n) Status And Control (FTM3_C0SC)	32	R/W	0000_0000h	36.3.6/751
4002_6010	Channel (n) Value (FTM3_C0V)	32	R/W	0000_0000h	36.3.7/754
4002_6014	Channel (n) Status And Control (FTM3_C1SC)	32	R/W	0000_0000h	36.3.6/751
4002_6018	Channel (n) Value (FTM3_C1V)	32	R/W	0000_0000h	36.3.7/754
4002_601C	Channel (n) Status And Control (FTM3_C2SC)	32	R/W	0000_0000h	36.3.6/751
4002_6020	Channel (n) Value (FTM3_C2V)	32	R/W	0000_0000h	36.3.7/754
4002_6024	Channel (n) Status And Control (FTM3_C3SC)	32	R/W	0000_0000h	36.3.6/751
4002_6028	Channel (n) Value (FTM3_C3V)	32	R/W	0000_0000h	36.3.7/754
4002_602C	Channel (n) Status And Control (FTM3_C4SC)	32	R/W	0000_0000h	36.3.6/751
4002_6030	Channel (n) Value (FTM3_C4V)	32	R/W	0000_0000h	36.3.7/754
4002_6034	Channel (n) Status And Control (FTM3_C5SC)	32	R/W	0000_0000h	36.3.6/751
4002_6038	Channel (n) Value (FTM3_C5V)	32	R/W	0000_0000h	36.3.7/754
4002_603C	Channel (n) Status And Control (FTM3_C6SC)	32	R/W	0000_0000h	36.3.6/751
4002_6040	Channel (n) Value (FTM3_C6V)	32	R/W	0000_0000h	36.3.7/754
4002_6044	Channel (n) Status And Control (FTM3_C7SC)	32	R/W	0000_0000h	36.3.6/751
4002_6048	Channel (n) Value (FTM3_C7V)	32	R/W	0000_0000h	36.3.7/754
4002_604C	Counter Initial Value (FTM3_CNTIN)	32	R/W	0000_0000h	36.3.8/754

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_6050	Capture And Compare Status (FTM3_STATUS)	32	R/W	0000_0000h	36.3.9/755
4002_6054	Features Mode Selection (FTM3_MODE)	32	R/W	0000_0004h	36.3.10/ 757
4002_6058	Synchronization (FTM3_SYNC)	32	R/W	0000_0000h	36.3.11/ 759
4002_605C	Initial State For Channels Output (FTM3_OUTINIT)	32	R/W	0000_0000h	36.3.12/ 761
4002_6060	Output Mask (FTM3_OUTMASK)	32	R/W	0000_0000h	36.3.13/ 762
4002_6064	Function For Linked Channels (FTM3_COMBINE)	32	R/W	0000_0000h	36.3.14/ 764
4002_6068	Deadtime Insertion Control (FTM3_DEADTIME)	32	R/W	0000_0000h	36.3.15/ 769
4002_606C	FTM External Trigger (FTM3_EXTTRIG)	32	R/W	0000_0000h	36.3.16/ 770
4002_6070	Channels Polarity (FTM3_POL)	32	R/W	0000_0000h	36.3.17/ 772
4002_6074	Fault Mode Status (FTM3_FMS)	32	R/W	0000_0000h	36.3.18/ 774
4002_6078	Input Capture Filter Control (FTM3_FILTER)	32	R/W	0000_0000h	36.3.19/ 776
4002_607C	Fault Control (FTM3_FLTCTRL)	32	R/W	0000_0000h	36.3.20/ 777
4002_6080	Quadrature Decoder Control And Status (FTM3_QDCTRL)	32	R/W	0000_0000h	36.3.21/ 780
4002_6084	Configuration (FTM3_CONF)	32	R/W	0000_0000h	36.3.22/ 782
4002_6088	FTM Fault Input Polarity (FTM3_FLTPOL)	32	R/W	0000_0000h	36.3.23/ 783
4002_608C	Synchronization Configuration (FTM3_SYNCONF)	32	R/W	0000_0000h	36.3.24/ 784
4002_6090	FTM Inverting Control (FTM3_INVCTRL)	32	R/W	0000_0000h	36.3.25/ 786
4002_6094	FTM Software Output Control (FTM3_SWOCTRL)	32	R/W	0000_0000h	36.3.26/ 787
4002_6098	FTM PWM Load (FTM3_PWMLOAD)	32	R/W	0000_0000h	36.3.27/ 790
4002_7000	Status And Control (FTM4_SC)	32	R/W	0000_0000h	36.3.3/748
4002_7004	Counter (FTM4_CNT)	32	R/W	0000_0000h	36.3.4/749
4002_7008	Modulo (FTM4_MOD)	32	R/W	0000_0000h	36.3.5/750
4002_700C	Channel (n) Status And Control (FTM4_C0SC)	32	R/W	0000_0000h	36.3.6/751
4002_7010	Channel (n) Value (FTM4_C0V)	32	R/W	0000_0000h	36.3.7/754
4002_7014	Channel (n) Status And Control (FTM4_C1SC)	32	R/W	0000_0000h	36.3.6/751

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_7018	Channel (n) Value (FTM4_C1V)	32	R/W	0000_0000h	36.3.7/754
4002_701C	Channel (n) Status And Control (FTM4_C2SC)	32	R/W	0000_0000h	36.3.6/751
4002_7020	Channel (n) Value (FTM4_C2V)	32	R/W	0000_0000h	36.3.7/754
4002_7024	Channel (n) Status And Control (FTM4_C3SC)	32	R/W	0000_0000h	36.3.6/751
4002_7028	Channel (n) Value (FTM4_C3V)	32	R/W	0000_0000h	36.3.7/754
4002_702C	Channel (n) Status And Control (FTM4_C4SC)	32	R/W	0000_0000h	36.3.6/751
4002_7030	Channel (n) Value (FTM4_C4V)	32	R/W	0000_0000h	36.3.7/754
4002_7034	Channel (n) Status And Control (FTM4_C5SC)	32	R/W	0000_0000h	36.3.6/751
4002_7038	Channel (n) Value (FTM4_C5V)	32	R/W	0000_0000h	36.3.7/754
4002_703C	Channel (n) Status And Control (FTM4_C6SC)	32	R/W	0000_0000h	36.3.6/751
4002_7040	Channel (n) Value (FTM4_C6V)	32	R/W	0000_0000h	36.3.7/754
4002_7044	Channel (n) Status And Control (FTM4_C7SC)	32	R/W	0000_0000h	36.3.6/751
4002_7048	Channel (n) Value (FTM4_C7V)	32	R/W	0000_0000h	36.3.7/754
4002_704C	Counter Initial Value (FTM4_CNTIN)	32	R/W	0000_0000h	36.3.8/754
4002_7050	Capture And Compare Status (FTM4_STATUS)	32	R/W	0000_0000h	36.3.9/755
4002_7054	Features Mode Selection (FTM4_MODE)	32	R/W	0000_0004h	36.3.10/ 757
4002_7058	Synchronization (FTM4_SYNC)	32	R/W	0000_0000h	36.3.11/ 759
4002_705C	Initial State For Channels Output (FTM4_OUTINIT)	32	R/W	0000_0000h	36.3.12/ 761
4002_7060	Output Mask (FTM4_OUTMASK)	32	R/W	0000_0000h	36.3.13/ 762
4002_7064	Function For Linked Channels (FTM4_COMBINE)	32	R/W	0000_0000h	36.3.14/ 764
4002_7068	Deadtime Insertion Control (FTM4_DEADTIME)	32	R/W	0000_0000h	36.3.15/ 769
4002_706C	FTM External Trigger (FTM4_EXTTRIG)	32	R/W	0000_0000h	36.3.16/ 770
4002_7070	Channels Polarity (FTM4_POL)	32	R/W	0000_0000h	36.3.17/ 772
4002_7074	Fault Mode Status (FTM4_FMS)	32	R/W	0000_0000h	36.3.18/ 774
4002_7078	Input Capture Filter Control (FTM4_FILTER)	32	R/W	0000_0000h	36.3.19/ 776
4002_707C	Fault Control (FTM4_FLTCTRL)	32	R/W	0000_0000h	36.3.20/ 777
4002_7080	Quadrature Decoder Control And Status (FTM4_QDCTRL)	32	R/W	0000_0000h	36.3.21/ 780
4002_7084	Configuration (FTM4_CONF)	32	R/W	0000_0000h	36.3.22/ 782

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_7088	FTM Fault Input Polarity (FTM4_FLTPOL)	32	R/W	0000_0000h	36.3.23/ 783
4002_708C	Synchronization Configuration (FTM4_SYNCONF)	32	R/W	0000_0000h	36.3.24/ 784
4002_7090	FTM Inverting Control (FTM4_INVCTRL)	32	R/W	0000_0000h	36.3.25/ 786
4002_7094	FTM Software Output Control (FTM4_SWOCTRL)	32	R/W	0000_0000h	36.3.26/ 787
4002_7098	FTM PWM Load (FTM4_PWMLOAD)	32	R/W	0000_0000h	36.3.27/ 790
4002_8000	Status And Control (FTM5_SC)	32	R/W	0000_0000h	36.3.3/748
4002_8004	Counter (FTM5_CNT)	32	R/W	0000_0000h	36.3.4/749
4002_8008	Modulo (FTM5_MOD)	32	R/W	0000_0000h	36.3.5/750
4002_800C	Channel (n) Status And Control (FTM5_C0SC)	32	R/W	0000_0000h	36.3.6/751
4002_8010	Channel (n) Value (FTM5_C0V)	32	R/W	0000_0000h	36.3.7/754
4002_8014	Channel (n) Status And Control (FTM5_C1SC)	32	R/W	0000_0000h	36.3.6/751
4002_8018	Channel (n) Value (FTM5_C1V)	32	R/W	0000_0000h	36.3.7/754
4002_801C	Channel (n) Status And Control (FTM5_C2SC)	32	R/W	0000_0000h	36.3.6/751
4002_8020	Channel (n) Value (FTM5_C2V)	32	R/W	0000_0000h	36.3.7/754
4002_8024	Channel (n) Status And Control (FTM5_C3SC)	32	R/W	0000_0000h	36.3.6/751
4002_8028	Channel (n) Value (FTM5_C3V)	32	R/W	0000_0000h	36.3.7/754
4002_802C	Channel (n) Status And Control (FTM5_C4SC)	32	R/W	0000_0000h	36.3.6/751
4002_8030	Channel (n) Value (FTM5_C4V)	32	R/W	0000_0000h	36.3.7/754
4002_8034	Channel (n) Status And Control (FTM5_C5SC)	32	R/W	0000_0000h	36.3.6/751
4002_8038	Channel (n) Value (FTM5_C5V)	32	R/W	0000_0000h	36.3.7/754
4002_803C	Channel (n) Status And Control (FTM5_C6SC)	32	R/W	0000_0000h	36.3.6/751
4002_8040	Channel (n) Value (FTM5_C6V)	32	R/W	0000_0000h	36.3.7/754
4002_8044	Channel (n) Status And Control (FTM5_C7SC)	32	R/W	0000_0000h	36.3.6/751
4002_8048	Channel (n) Value (FTM5_C7V)	32	R/W	0000_0000h	36.3.7/754
4002_804C	Counter Initial Value (FTM5_CNTIN)	32	R/W	0000_0000h	36.3.8/754
4002_8050	Capture And Compare Status (FTM5_STATUS)	32	R/W	0000_0000h	36.3.9/755
4002_8054	Features Mode Selection (FTM5_MODE)	32	R/W	0000_0004h	36.3.10/ 757
4002_8058	Synchronization (FTM5_SYNC)	32	R/W	0000_0000h	36.3.11/ 759
4002_805C	Initial State For Channels Output (FTM5_OUTINIT)	32	R/W	0000_0000h	36.3.12/ 761
4002_8060	Output Mask (FTM5_OUTMASK)	32	R/W	0000_0000h	36.3.13/ 762
4002_8064	Function For Linked Channels (FTM5_COMBINE)	32	R/W	0000_0000h	36.3.14/ 764

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_8068	Deadtime Insertion Control (FTM5_DEADTIME)	32	R/W	0000_0000h	36.3.15/ 769
4002_806C	FTM External Trigger (FTM5_EXTTRIG)	32	R/W	0000_0000h	36.3.16/ 770
4002_8070	Channels Polarity (FTM5_POL)	32	R/W	0000_0000h	36.3.17/ 772
4002_8074	Fault Mode Status (FTM5_FMS)	32	R/W	0000_0000h	36.3.18/ 774
4002_8078	Input Capture Filter Control (FTM5_FILTER)	32	R/W	0000_0000h	36.3.19/ 776
4002_807C	Fault Control (FTM5_FLTCTRL)	32	R/W	0000_0000h	36.3.20/ 777
4002_8080	Quadrature Decoder Control And Status (FTM5_QDCTRL)	32	R/W	0000_0000h	36.3.21/ 780
4002_8084	Configuration (FTM5_CONF)	32	R/W	0000_0000h	36.3.22/ 782
4002_8088	FTM Fault Input Polarity (FTM5_FLTPOL)	32	R/W	0000_0000h	36.3.23/ 783
4002_808C	Synchronization Configuration (FTM5_SYNCONF)	32	R/W	0000_0000h	36.3.24/ 784
4002_8090	FTM Inverting Control (FTM5_INVCTRL)	32	R/W	0000_0000h	36.3.25/ 786
4002_8094	FTM Software Output Control (FTM5_SWOCTRL)	32	R/W	0000_0000h	36.3.26/ 787
4002_8098	FTM PWM Load (FTM5_PWMLOAD)	32	R/W	0000_0000h	36.3.27/ 790
4003_8000	Status And Control (FTM0_SC)	32	R/W	0000_0000h	36.3.3/748
4003_8004	Counter (FTM0_CNT)	32	R/W	0000_0000h	36.3.4/749
4003_8008	Modulo (FTM0_MOD)	32	R/W	0000_0000h	36.3.5/750
4003_800C	Channel (n) Status And Control (FTM0_C0SC)	32	R/W	0000_0000h	36.3.6/751
4003_8010	Channel (n) Value (FTM0_C0V)	32	R/W	0000_0000h	36.3.7/754
4003_8014	Channel (n) Status And Control (FTM0_C1SC)	32	R/W	0000_0000h	36.3.6/751
4003_8018	Channel (n) Value (FTM0_C1V)	32	R/W	0000_0000h	36.3.7/754
4003_801C	Channel (n) Status And Control (FTM0_C2SC)	32	R/W	0000_0000h	36.3.6/751
4003_8020	Channel (n) Value (FTM0_C2V)	32	R/W	0000_0000h	36.3.7/754
4003_8024	Channel (n) Status And Control (FTM0_C3SC)	32	R/W	0000_0000h	36.3.6/751
4003_8028	Channel (n) Value (FTM0_C3V)	32	R/W	0000_0000h	36.3.7/754
4003_802C	Channel (n) Status And Control (FTM0_C4SC)	32	R/W	0000_0000h	36.3.6/751
4003_8030	Channel (n) Value (FTM0_C4V)	32	R/W	0000_0000h	36.3.7/754
4003_8034	Channel (n) Status And Control (FTM0_C5SC)	32	R/W	0000_0000h	36.3.6/751
4003_8038	Channel (n) Value (FTM0_C5V)	32	R/W	0000_0000h	36.3.7/754
4003_803C	Channel (n) Status And Control (FTM0_C6SC)	32	R/W	0000_0000h	36.3.6/751

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_8040	Channel (n) Value (FTM0_C6V)	32	R/W	0000_0000h	36.3.7/754
4003_8044	Channel (n) Status And Control (FTM0_C7SC)	32	R/W	0000_0000h	36.3.6/751
4003_8048	Channel (n) Value (FTM0_C7V)	32	R/W	0000_0000h	36.3.7/754
4003_804C	Counter Initial Value (FTM0_CNTIN)	32	R/W	0000_0000h	36.3.8/754
4003_8050	Capture And Compare Status (FTM0_STATUS)	32	R/W	0000_0000h	36.3.9/755
4003_8054	Features Mode Selection (FTM0_MODE)	32	R/W	0000_0004h	36.3.10/ 757
4003_8058	Synchronization (FTM0_SYNC)	32	R/W	0000_0000h	36.3.11/ 759
4003_805C	Initial State For Channels Output (FTM0_OUTINIT)	32	R/W	0000_0000h	36.3.12/ 761
4003_8060	Output Mask (FTM0_OUTMASK)	32	R/W	0000_0000h	36.3.13/ 762
4003_8064	Function For Linked Channels (FTM0_COMBINE)	32	R/W	0000_0000h	36.3.14/ 764
4003_8068	Deadtime Insertion Control (FTM0_DEADTIME)	32	R/W	0000_0000h	36.3.15/ 769
4003_806C	FTM External Trigger (FTM0_EXTTRIG)	32	R/W	0000_0000h	36.3.16/ 770
4003_8070	Channels Polarity (FTM0_POL)	32	R/W	0000_0000h	36.3.17/ 772
4003_8074	Fault Mode Status (FTM0_FMS)	32	R/W	0000_0000h	36.3.18/ 774
4003_8078	Input Capture Filter Control (FTM0_FILTER)	32	R/W	0000_0000h	36.3.19/ 776
4003_807C	Fault Control (FTM0_FLTCTRL)	32	R/W	0000_0000h	36.3.20/ 777
4003_8080	Quadrature Decoder Control And Status (FTM0_QDCTRL)	32	R/W	0000_0000h	36.3.21/ 780
4003_8084	Configuration (FTM0_CONF)	32	R/W	0000_0000h	36.3.22/ 782
4003_8088	FTM Fault Input Polarity (FTM0_FLTPOL)	32	R/W	0000_0000h	36.3.23/ 783
4003_808C	Synchronization Configuration (FTM0_SYNCONF)	32	R/W	0000_0000h	36.3.24/ 784
4003_8090	FTM Inverting Control (FTM0_INVCTRL)	32	R/W	0000_0000h	36.3.25/ 786
4003_8094	FTM Software Output Control (FTM0_SWOCTRL)	32	R/W	0000_0000h	36.3.26/ 787
4003_8098	FTM PWM Load (FTM0_PWMLOAD)	32	R/W	0000_0000h	36.3.27/ 790
4003_9000	Status And Control (FTM1_SC)	32	R/W	0000_0000h	36.3.3/748
4003_9004	Counter (FTM1_CNT)	32	R/W	0000_0000h	36.3.4/749

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_9008	Modulo (FTM1_MOD)	32	R/W	0000_0000h	36.3.5/750
4003_900C	Channel (n) Status And Control (FTM1_C0SC)	32	R/W	0000_0000h	36.3.6/751
4003_9010	Channel (n) Value (FTM1_C0V)	32	R/W	0000_0000h	36.3.7/754
4003_9014	Channel (n) Status And Control (FTM1_C1SC)	32	R/W	0000_0000h	36.3.6/751
4003_9018	Channel (n) Value (FTM1_C1V)	32	R/W	0000_0000h	36.3.7/754
4003_901C	Channel (n) Status And Control (FTM1_C2SC)	32	R/W	0000_0000h	36.3.6/751
4003_9020	Channel (n) Value (FTM1_C2V)	32	R/W	0000_0000h	36.3.7/754
4003_9024	Channel (n) Status And Control (FTM1_C3SC)	32	R/W	0000_0000h	36.3.6/751
4003_9028	Channel (n) Value (FTM1_C3V)	32	R/W	0000_0000h	36.3.7/754
4003_902C	Channel (n) Status And Control (FTM1_C4SC)	32	R/W	0000_0000h	36.3.6/751
4003_9030	Channel (n) Value (FTM1_C4V)	32	R/W	0000_0000h	36.3.7/754
4003_9034	Channel (n) Status And Control (FTM1_C5SC)	32	R/W	0000_0000h	36.3.6/751
4003_9038	Channel (n) Value (FTM1_C5V)	32	R/W	0000_0000h	36.3.7/754
4003_903C	Channel (n) Status And Control (FTM1_C6SC)	32	R/W	0000_0000h	36.3.6/751
4003_9040	Channel (n) Value (FTM1_C6V)	32	R/W	0000_0000h	36.3.7/754
4003_9044	Channel (n) Status And Control (FTM1_C7SC)	32	R/W	0000_0000h	36.3.6/751
4003_9048	Channel (n) Value (FTM1_C7V)	32	R/W	0000_0000h	36.3.7/754
4003_904C	Counter Initial Value (FTM1_CNTIN)	32	R/W	0000_0000h	36.3.8/754
4003_9050	Capture And Compare Status (FTM1_STATUS)	32	R/W	0000_0000h	36.3.9/755
4003_9054	Features Mode Selection (FTM1_MODE)	32	R/W	0000_0004h	36.3.10/ 757
4003_9058	Synchronization (FTM1_SYNC)	32	R/W	0000_0000h	36.3.11/ 759
4003_905C	Initial State For Channels Output (FTM1_OUTINIT)	32	R/W	0000_0000h	36.3.12/ 761
4003_9060	Output Mask (FTM1_OUTMASK)	32	R/W	0000_0000h	36.3.13/ 762
4003_9064	Function For Linked Channels (FTM1_COMBINE)	32	R/W	0000_0000h	36.3.14/ 764
4003_9068	Deadtime Insertion Control (FTM1_DEADTIME)	32	R/W	0000_0000h	36.3.15/ 769
4003_906C	FTM External Trigger (FTM1_EXTTRIG)	32	R/W	0000_0000h	36.3.16/ 770
4003_9070	Channels Polarity (FTM1_POL)	32	R/W	0000_0000h	36.3.17/ 772
4003_9074	Fault Mode Status (FTM1_FMS)	32	R/W	0000_0000h	36.3.18/ 774
4003_9078	Input Capture Filter Control (FTM1_FILTER)	32	R/W	0000_0000h	36.3.19/ 776
4003_907C	Fault Control (FTM1_FLTCTRL)	32	R/W	0000_0000h	36.3.20/ 777

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_9080	Quadrature Decoder Control And Status (FTM1_QDCTRL)	32	R/W	0000_0000h	36.3.21/ 780
4003_9084	Configuration (FTM1_CONF)	32	R/W	0000_0000h	36.3.22/ 782
4003_9088	FTM Fault Input Polarity (FTM1_FLTPOL)	32	R/W	0000_0000h	36.3.23/ 783
4003_908C	Synchronization Configuration (FTM1_SYNCONF)	32	R/W	0000_0000h	36.3.24/ 784
4003_9090	FTM Inverting Control (FTM1_INVCTRL)	32	R/W	0000_0000h	36.3.25/ 786
4003_9094	FTM Software Output Control (FTM1_SWOCTRL)	32	R/W	0000_0000h	36.3.26/ 787
4003_9098	FTM PWM Load (FTM1_PWMLOAD)	32	R/W	0000_0000h	36.3.27/ 790
4003_A000	Status And Control (FTM2_SC)	32	R/W	0000_0000h	36.3.3/748
4003_A004	Counter (FTM2_CNT)	32	R/W	0000_0000h	36.3.4/749
4003_A008	Modulo (FTM2_MOD)	32	R/W	0000_0000h	36.3.5/750
4003_A00C	Channel (n) Status And Control (FTM2_C0SC)	32	R/W	0000_0000h	36.3.6/751
4003_A010	Channel (n) Value (FTM2_C0V)	32	R/W	0000_0000h	36.3.7/754
4003_A014	Channel (n) Status And Control (FTM2_C1SC)	32	R/W	0000_0000h	36.3.6/751
4003_A018	Channel (n) Value (FTM2_C1V)	32	R/W	0000_0000h	36.3.7/754
4003_A01C	Channel (n) Status And Control (FTM2_C2SC)	32	R/W	0000_0000h	36.3.6/751
4003_A020	Channel (n) Value (FTM2_C2V)	32	R/W	0000_0000h	36.3.7/754
4003_A024	Channel (n) Status And Control (FTM2_C3SC)	32	R/W	0000_0000h	36.3.6/751
4003_A028	Channel (n) Value (FTM2_C3V)	32	R/W	0000_0000h	36.3.7/754
4003_A02C	Channel (n) Status And Control (FTM2_C4SC)	32	R/W	0000_0000h	36.3.6/751
4003_A030	Channel (n) Value (FTM2_C4V)	32	R/W	0000_0000h	36.3.7/754
4003_A034	Channel (n) Status And Control (FTM2_C5SC)	32	R/W	0000_0000h	36.3.6/751
4003_A038	Channel (n) Value (FTM2_C5V)	32	R/W	0000_0000h	36.3.7/754
4003_A03C	Channel (n) Status And Control (FTM2_C6SC)	32	R/W	0000_0000h	36.3.6/751
4003_A040	Channel (n) Value (FTM2_C6V)	32	R/W	0000_0000h	36.3.7/754
4003_A044	Channel (n) Status And Control (FTM2_C7SC)	32	R/W	0000_0000h	36.3.6/751
4003_A048	Channel (n) Value (FTM2_C7V)	32	R/W	0000_0000h	36.3.7/754
4003_A04C	Counter Initial Value (FTM2_CNTIN)	32	R/W	0000_0000h	36.3.8/754
4003_A050	Capture And Compare Status (FTM2_STATUS)	32	R/W	0000_0000h	36.3.9/755
4003_A054	Features Mode Selection (FTM2_MODE)	32	R/W	0000_0004h	36.3.10/ 757
4003_A058	Synchronization (FTM2_SYNC)	32	R/W	0000_0000h	36.3.11/ 759
4003_A05C	Initial State For Channels Output (FTM2_OUTINIT)	32	R/W	0000_0000h	36.3.12/ 761

Table continues on the next page...

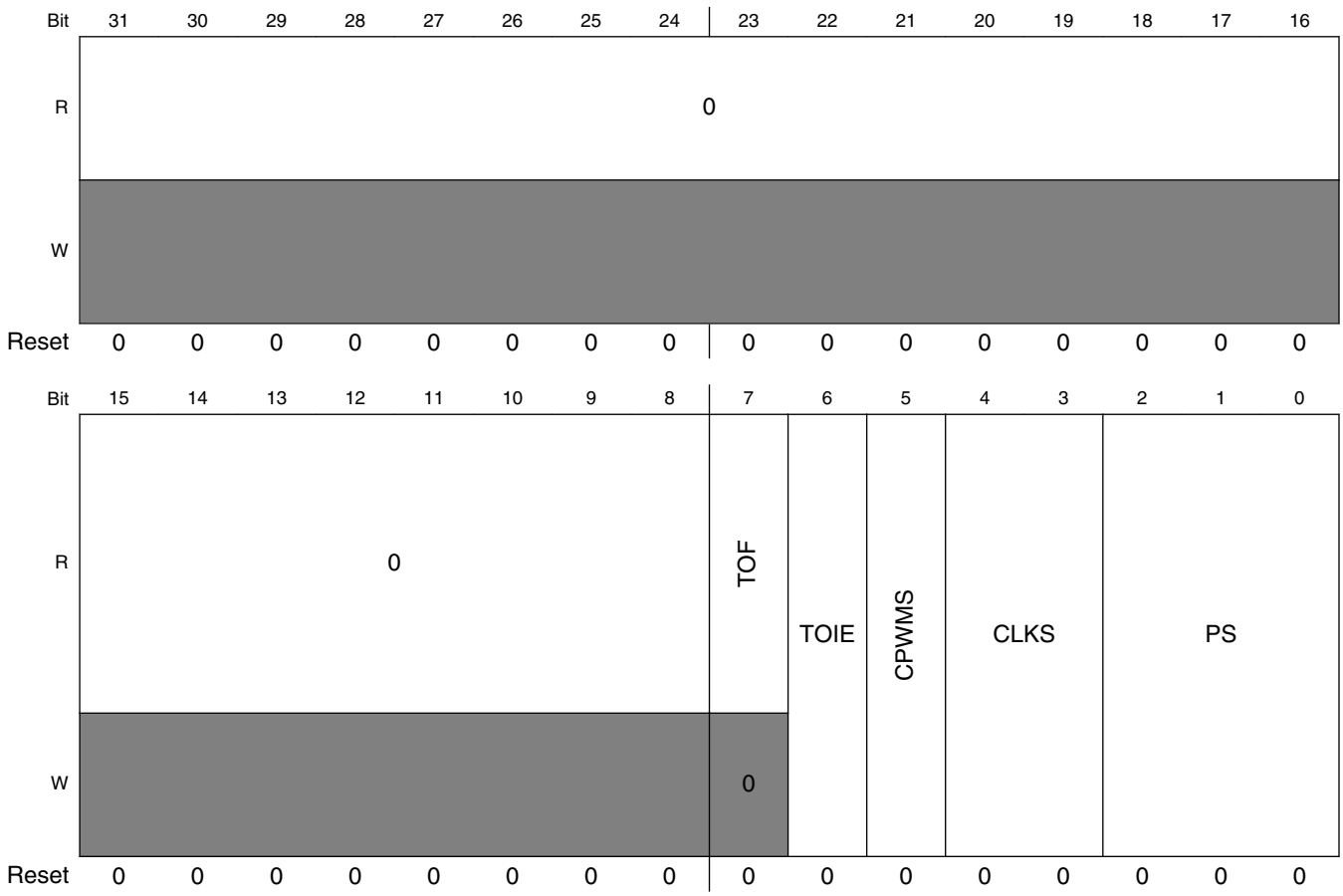
FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_A060	Output Mask (FTM2_OUTMASK)	32	R/W	0000_0000h	36.3.13/ 762
4003_A064	Function For Linked Channels (FTM2_COMBINE)	32	R/W	0000_0000h	36.3.14/ 764
4003_A068	Deadtime Insertion Control (FTM2_DEADTIME)	32	R/W	0000_0000h	36.3.15/ 769
4003_A06C	FTM External Trigger (FTM2_EXTTRIG)	32	R/W	0000_0000h	36.3.16/ 770
4003_A070	Channels Polarity (FTM2_POL)	32	R/W	0000_0000h	36.3.17/ 772
4003_A074	Fault Mode Status (FTM2_FMS)	32	R/W	0000_0000h	36.3.18/ 774
4003_A078	Input Capture Filter Control (FTM2_FILTER)	32	R/W	0000_0000h	36.3.19/ 776
4003_A07C	Fault Control (FTM2_FLTCTRL)	32	R/W	0000_0000h	36.3.20/ 777
4003_A080	Quadrature Decoder Control And Status (FTM2_QDCTRL)	32	R/W	0000_0000h	36.3.21/ 780
4003_A084	Configuration (FTM2_CONF)	32	R/W	0000_0000h	36.3.22/ 782
4003_A088	FTM Fault Input Polarity (FTM2_FLTPOL)	32	R/W	0000_0000h	36.3.23/ 783
4003_A08C	Synchronization Configuration (FTM2_SYNCONF)	32	R/W	0000_0000h	36.3.24/ 784
4003_A090	FTM Inverting Control (FTM2_INVCTRL)	32	R/W	0000_0000h	36.3.25/ 786
4003_A094	FTM Software Output Control (FTM2_SWOCTRL)	32	R/W	0000_0000h	36.3.26/ 787
4003_A098	FTM PWM Load (FTM2_PWMLOAD)	32	R/W	0000_0000h	36.3.27/ 790

36.3.3 Status And Control (FTMx_SC)

SC contains the overflow status flag and control bits used to configure the interrupt enable, FTM configuration, clock source, and prescaler factor. These controls relate to all channels within this module.

Address: Base address + 0h offset



FTMx_SC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TOF	Timer Overflow Flag Set by hardware when the FTM counter passes the value in the MOD register. The TOF bit is cleared by reading the SC register while TOF is set and then writing a 0 to TOF bit. Writing a 1 to TOF has no effect. If another FTM overflow occurs between the read and write operations, the write operation has no effect; therefore, TOF remains set indicating an overflow has occurred. In this case, a TOF interrupt request is not lost due to the clearing sequence for a previous TOF. 0 FTM counter has not overflowed. 1 FTM counter has overflowed.

Table continues on the next page...

FTMx_SC field descriptions (continued)

Field	Description
6 TOIE	Timer Overflow Interrupt Enable Enables FTM overflow interrupts. 0 Disable TOF interrupts. Use software polling. 1 Enable TOF interrupts. An interrupt is generated when TOF equals one.
5 CPWMS	Center-Aligned PWM Select Selects CPWM mode. This mode configures the FTM to operate in Up-Down Counting mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 FTM counter operates in Up Counting mode. 1 FTM counter operates in Up-Down Counting mode.
4–3 CLKS	Clock Source Selection Selects FTM counter clock sources. This field is write protected. It can be written only when MODE[WPDIS] = 1. 00 No clock selected. This in effect disables the FTM counter. 01 System clock 10 Fixed frequency clock 11 External clock
PS	Prescale Factor Selection Selects one of 8 division factors for the clock source selected by CLKS. The new prescaler factor affects the clock source on the next system clock cycle after the new value is updated into the register bits. This field is write protected. It can be written only when MODE[WPDIS] = 1. 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

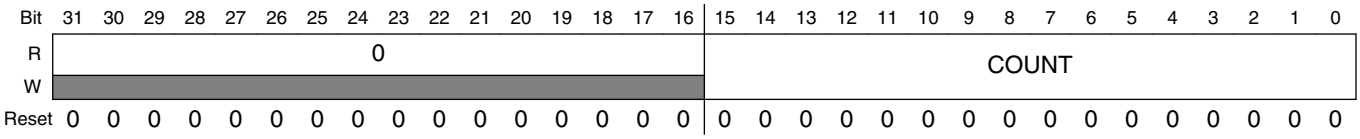
36.3.4 Counter (FTMx_CNT)

The CNT register contains the FTM counter value.

Reset clears the CNT register. Writing any value to COUNT updates the counter with its initial value, CNTIN.

When BDM is active, the FTM counter is frozen. This is the value that you may read.

Address: Base address + 4h offset



FTMx_CNT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Counter Value

36.3.5 Modulo (FTMx_MOD)

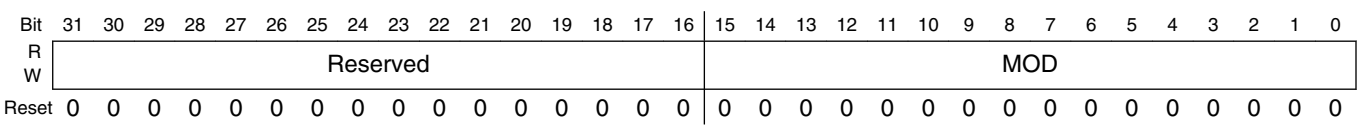
The Modulo register contains the modulo value for the FTM counter. After the FTM counter reaches the modulo value, the overflow flag (TOF) becomes set at the next clock, and the next value of FTM counter depends on the selected counting method; see Counter.

Writing to the MOD register latches the value into a buffer. The MOD register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the SC register whether BDM is active or not.

Initialize the FTM counter, by writing to CNT, before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

Address: Base address + 8h offset



FTMx_MOD field descriptions

Field	Description
31–16 Reserved	This field is reserved.
MOD	Modulo Value

36.3.6 Channel (n) Status And Control (FTM*x*_C*n*SC)

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

Table 36-2. Mode, edge, and level selection

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	X	X	XX	00	Pin not used for FTM—revert the channel pin to general purpose I/O or other peripheral control	
0	0	0	00	01	Input Capture	Capture on Rising Edge Only
				10		Capture on Falling Edge Only
				11		Capture on Rising or Falling Edge
			01	01	Output Compare	Toggle Output on match
				10		Clear Output on match
				11		Set Output on match
			1X	10	Edge-Aligned PWM	High-true pulses (clear Output on match)
				X1		Low-true pulses (set Output on match)
		1	XX	10	Center-Aligned PWM	High-true pulses (clear Output on match-up)
				X1		Low-true pulses (set Output on match-up)
	1	0	XX	10	Combine PWM	High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
				X1		Low-true pulses (clear on channel (n) match, and set on channel (n +1) match)

Table continues on the next page...

Table 36-2. Mode, edge, and level selection (continued)

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
1	0	0	X0	See the following table (Table 36-3).	Dual Edge Capture	One-Shot Capture mode
			X1			Continuous Capture mode

Table 36-3. Dual Edge Capture mode — edge polarity selection

ELSnB	ELSnA	Channel Port Enable	Detected Edges
0	0	Disabled	No edge
0	1	Enabled	Rising edge
1	0	Enabled	Falling edge
1	1	Enabled	Rising and falling edges

Address: Base address + Ch offset + (8d × 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															
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								CHF	CHIE	MSB	MSA	ELSB	ELSA	ICRST	DMA
W									0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_C*n*SC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CHF	Channel Flag Set by hardware when an event occurs on the channel. CHF is cleared by reading the CSC register while CHnF is set and then writing a 0 to the CHF bit. Writing a 1 to CHF has no effect. If another event occurs between the read and write operations, the write operation has no effect; therefore, CHF remains set indicating an event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF. 0 No channel event has occurred. 1 A channel event has occurred.
6 CHIE	Channel Interrupt Enable Enables channel interrupts. 0 Disable channel interrupts. Use software polling. 1 Enable channel interrupts.
5 MSB	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 36-2 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
4 MSA	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 36-2 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
3 ELSB	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. See Table 36-2 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
2 ELSA	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. See Table 36-2 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
1 ICRST	FTM counter reset by the selected input capture event. FTM counter reset is driven by the selected event of the channel (n) in the Input Capture mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 FTM counter is not reset when the selected channel (n) input event is detected. 1 FTM counter is reset when the selected channel (n) input event is detected.
0 DMA	DMA Enable Enables DMA transfers for the channel. 0 Disable DMA transfers. 1 Enable DMA transfers.

36.3.7 Channel (n) Value (FTMx_CnV)

These registers contain the captured FTM counter value for the input modes or the match value for the output modes.

In Input Capture, Capture Test, and Dual Edge Capture modes, any write to a CnV register is ignored.

In output modes, writing to a CnV register latches the value into a buffer. A CnV register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the CnSC register whether BDM mode is active or not.

Address: Base address + 10h offset + (8d × 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	0																VAL															
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	0	0	0

FTMx_CnV field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
VAL	Channel Value Captured FTM counter value of the input modes or the match value for the output modes

36.3.8 Counter Initial Value (FTMx_CNTIN)

The Counter Initial Value register contains the initial value for the FTM counter.

Writing to the CNTIN register latches the value into a buffer. The CNTIN register is updated with the value of its write buffer according to Registers updated from write buffers.

When the FTM clock is initially selected, by writing a non-zero value to the CLKS bits, the FTM counter starts with the value 0x0000. To avoid this behavior, before the first write to select the FTM clock, write the new value to the the CNTIN register and then initialize the FTM counter by writing any value to the CNT register.

Address: Base address + 4Ch 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 W	Reserved																INIT															
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

FTMx_CNTIN field descriptions

Field	Description
31–16 Reserved	This field is reserved.
INIT	Initial Value Of The FTM Counter

36.3.9 Capture And Compare Status (FTMx_STATUS)

The STATUS register contains a copy of the status flag CHnF bit in CnSC for each FTM channel for software convenience.

Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by reading STATUS followed by writing 0x00 to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. CHnF is cleared by reading STATUS while CHnF is set and then writing a 0 to the CHnF bit. Writing a 1 to CHnF has no effect.

If another event occurs between the read and write operations, the write operation has no effect; therefore, CHnF remains set indicating an event has occurred. In this case, a CHnF interrupt request is not lost due to the clearing sequence for a previous CHnF.

Address: Base address + 50h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W									0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_STATUS field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CH7F	Channel 7 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
6 CH6F	Channel 6 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
5 CH5F	Channel 5 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
4 CH4F	Channel 4 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
3 CH3F	Channel 3 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
2 CH2F	Channel 2 Flag See the register description.

Table continues on the next page...

FTMx_STATUS field descriptions (continued)

Field	Description
	0 No channel event has occurred. 1 A channel event has occurred.
1 CH1F	Channel 1 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
0 CH0F	Channel 0 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.

36.3.10 Features Mode Selection (FTMx_MODE)

This register contains the global enable bit for FTM-specific features and the control bits used to configure:

- Fault control mode and interrupt
- Capture Test mode
- PWM synchronization
- Write protection
- Channel output initialization

These controls relate to all channels within this module.

Address: Base address + 54h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								FAULTIE	FAULTM		CAPTEST	PWMSYNC	WPDIS	INIT	FTMEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

FTMx_MODE field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 FAULTIE	Fault Interrupt Enable Enables the generation of an interrupt when a fault is detected by FTM and the FTM fault control is enabled. 0 Fault control interrupt is disabled. 1 Fault control interrupt is enabled.
6–5 FAULTM	Fault Control Mode Defines the FTM fault control mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 00 Fault control is disabled for all channels. 01 Fault control is enabled for even channels only (channels 0, 2, 4, and 6), and the selected mode is the manual fault clearing. 10 Fault control is enabled for all channels, and the selected mode is the manual fault clearing. 11 Fault control is enabled for all channels, and the selected mode is the automatic fault clearing.
4 CAPTEST	Capture Test Mode Enable Enables the capture test mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Capture test mode is disabled. 1 Capture test mode is enabled.
3 PWMSYNC	PWM Synchronization Mode Selects which triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization. See PWM synchronization. The PWMSYNC bit configures the synchronization when SYNCMODE is 0. 0 No restrictions. Software and hardware triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization. 1 Software trigger can only be used by MOD and CnV synchronization, and hardware triggers can only be used by OUTMASK and FTM counter synchronization.
2 WPDIS	Write Protection Disable When write protection is enabled (WPDIS = 0), write protected bits cannot be written. When write protection is disabled (WPDIS = 1), write protected bits can be written. The WPDIS bit is the negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect. 0 Write protection is enabled. 1 Write protection is disabled.
1 INIT	Initialize The Channels Output When a 1 is written to INIT bit the channels output is initialized according to the state of their corresponding bit in the OUTINIT register. Writing a 0 to INIT bit has no effect. The INIT bit is always read as 0.
0 FTMEN	FTM Enable This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

FTMx_MODE field descriptions (continued)

Field	Description
	0 TPM compatibility. Free running counter and synchronization compatible with TPM. 1 Free running counter and synchronization are different from TPM behavior.

36.3.11 Synchronization (FTMx_SYNC)

This register configures the PWM synchronization.

A synchronization event can perform the synchronized update of MOD, CV, and OUTMASK registers with the value of their write buffer and the FTM counter initialization.

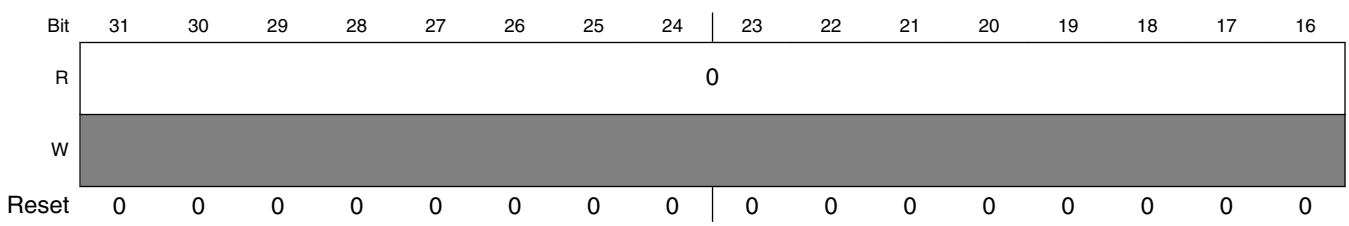
NOTE

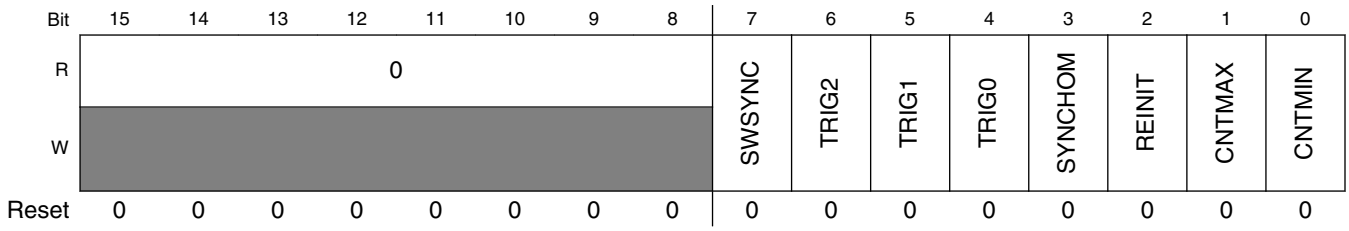
The software trigger, SWSYNC bit, and hardware triggers TRIG0, TRIG1, and TRIG2 bits have a potential conflict if used together when SYNCMODE = 0. Use only hardware or software triggers but not both at the same time, otherwise unpredictable behavior is likely to happen.

The selection of the loading point, CNTMAX and CNTMIN bits, is intended to provide the update of MOD, CNTIN, and CnV registers across all enabled channels simultaneously. The use of the loading point selection together with SYNCMODE = 0 and hardware trigger selection, TRIG0, TRIG1, or TRIG2 bits, is likely to result in unpredictable behavior.

The synchronization event selection also depends on the PWMSYNC (MODE register) and SYNCMODE (SYNCONF register) bits. See PWM synchronization.

Address: Base address + 58h offset





FTMx_SYNC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SWSYNC	PWM Synchronization Software Trigger Selects the software trigger as the PWM synchronization trigger. The software trigger happens when a 1 is written to SWSYNC bit. 0 Software trigger is not selected. 1 Software trigger is selected.
6 TRIG2	PWM Synchronization Hardware Trigger 2 Enables hardware trigger 2 to the PWM synchronization. Hardware trigger 2 happens when a rising edge is detected at the trigger 2 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
5 TRIG1	PWM Synchronization Hardware Trigger 1 Enables hardware trigger 1 to the PWM synchronization. Hardware trigger 1 happens when a rising edge is detected at the trigger 1 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
4 TRIG0	PWM Synchronization Hardware Trigger 0 Enables hardware trigger 0 to the PWM synchronization. Hardware trigger 0 occurs when a rising edge is detected at the trigger 0 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
3 SYNCHOM	Output Mask Synchronization Selects when the OUTMASK register is updated with the value of its buffer. 0 OUTMASK register is updated with the value of its buffer in all rising edges of the system clock. 1 OUTMASK register is updated with the value of its buffer only by the PWM synchronization.
2 REINIT	FTM Counter Reinitialization By Synchronization (FTM counter synchronization) Determines if the FTM counter is reinitialized when the selected trigger for the synchronization is detected. The REINIT bit configures the synchronization when SYNCMODE is zero. 0 FTM counter continues to count normally. 1 FTM counter is updated with its initial value when the selected trigger is detected.

Table continues on the next page...

FTMx_SYNC field descriptions (continued)

Field	Description
1 CNTMAX	Maximum Loading Point Enable Selects the maximum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMAX is 1, the selected loading point is when the FTM counter reaches its maximum value (MOD register). 0 The maximum loading point is disabled. 1 The maximum loading point is enabled.
0 CNTMIN	Minimum Loading Point Enable Selects the minimum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMIN is one, the selected loading point is when the FTM counter reaches its minimum value (CNTIN register). 0 The minimum loading point is disabled. 1 The minimum loading point is enabled.

36.3.12 Initial State For Channels Output (FTMx_OUTINIT)

Address: Base address + 5Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								CH7OI	CH6OI	CH5OI	CH4OI	CH3OI	CH2OI	CH1OI	CH0OI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_OUTINIT field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CH7OI	Channel 7 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
6 CH6OI	Channel 6 Output Initialization Value

Table continues on the next page...

FTMx_OUTINIT field descriptions (continued)

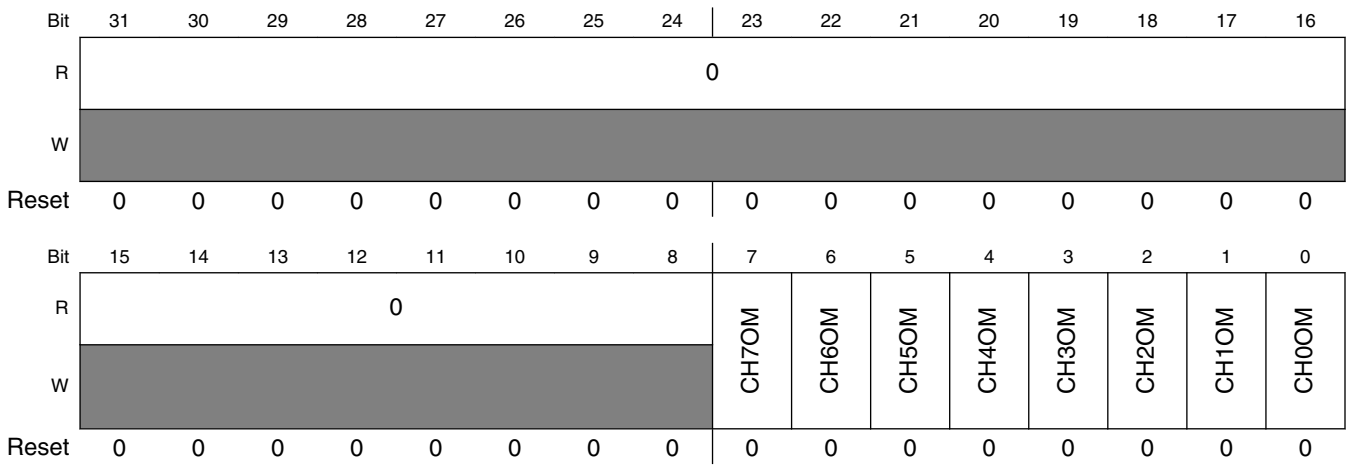
Field	Description
	Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
5 CH5OI	Channel 5 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
4 CH4OI	Channel 4 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
3 CH3OI	Channel 3 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
2 CH2OI	Channel 2 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
1 CH1OI	Channel 1 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
0 CH0OI	Channel 0 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.

36.3.13 Output Mask (FTMx_OUTMASK)

This register provides a mask for each FTM channel. The mask of a channel determines if its output responds, that is, it is masked or not, when a match occurs. This feature is used for BLDC control where the PWM signal is presented to an electric motor at specific times to provide electronic commutation.

Any write to the OUTMASK register, stores the value in its write buffer. The register is updated with the value of its write buffer according to PWM synchronization.

Address: Base address + 60h offset



FTMx_OUTMASK field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CH7OM	Channel 7 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
6 CH6OM	Channel 6 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
5 CH5OM	Channel 5 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
4 CH4OM	Channel 4 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
3 CH3OM	Channel 3 Output Mask Defines if the channel output is masked or unmasked.

Table continues on the next page...

FTMx_OUTMASK field descriptions (continued)

Field	Description
	0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
2 CH2OM	Channel 2 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
1 CH1OM	Channel 1 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
0 CH0OM	Channel 0 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.

36.3.14 Function For Linked Channels (FTMx_COMBINE)

This register contains the control bits used to configure the fault control, synchronization, deadtime insertion, Dual Edge Capture mode, Complementary, and Combine mode for each pair of channels (n) and (n+1), where n equals 0, 2, 4, and 6.

Address: Base address + 64h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	FAULTEN3	SYNCEN3	DTEN3	DECAP3	DECAPEN3	COMP3	COMBINE3	0	FAULTEN2	SYNCEN2	DTEN2	DECAP2	DECAPEN2	COMP2	COMBINE2
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	FAULTEN1	SYNCEN1	DTEN1	DECAP1	DECAPEN1	COMP1	COMBINE1	0	FAULTEN0	SYNCEN0	DTEN0	DECAP0	DECAPEN0	COMP0	COMBINE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_COMBINE field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FAULTEN3	Fault Control Enable For n = 6 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
29 SYNCEN3	Synchronization Enable For n = 6 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
28 DTEN3	Deadtime Enable For n = 6 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
27 DECAP3	Dual Edge Capture Mode Captures For n = 6 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
26 DECAPEN3	Dual Edge Capture Mode Enable For n = 6 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 36-2. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
25 COMP3	Complement Of Channel (n) for n = 6 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
24 COMBINE3	Combine Channels For n = 6 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 FAULTEN2	Fault Control Enable For n = 4 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
21 SYNCEN2	Synchronization Enable For n = 4 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
20 DTEN2	Deadtime Enable For n = 4 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
19 DECAP2	Dual Edge Capture Mode Captures For n = 4 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
18 DECAPEN2	Dual Edge Capture Mode Enable For n = 4 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 36-2. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
17 COMP2	Complement Of Channel (n) For n = 4

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
16 COMBINE2	Combine Channels For n = 4 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 FAULTEN1	Fault Control Enable For n = 2 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
13 SYNCEN1	Synchronization Enable For n = 2 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
12 DTEN1	Deadtime Enable For n = 2 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
11 DECAP1	Dual Edge Capture Mode Captures For n = 2 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if Dual Edge Capture – One-Shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
10 DECAPEN1	Dual Edge Capture Mode Enable For n = 2 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 36-2.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
9 COMP1	Complement Of Channel (n) For n = 2 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
8 COMBINE1	Combine Channels For n = 2 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FAULTEN0	Fault Control Enable For n = 0 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
5 SYNCEN0	Synchronization Enable For n = 0 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
4 DTEN0	Deadtime Enable For n = 0 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
3 DECAP0	Dual Edge Capture Mode Captures For n = 0 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
2 DECAPEN0	Dual Edge Capture Mode Enable For n = 0 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 36-2. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
1 COMP0	Complement Of Channel (n) For n = 0 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
0 COMBINE0	Combine Channels For n = 0 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.

36.3.15 Deadtime Insertion Control (FTMx_DEADTIME)

This register selects the deadtime prescaler factor and deadtime value. All FTM channels use this clock prescaler and this deadtime value for the deadtime insertion.

Address: Base address + 68h 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																								DTPS		DTVAL					
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	0	0	0

FTMx_DEADTIME field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 DTPS	Deadtime Prescaler Value Selects the division factor of the system clock. This prescaled clock is used by the deadtime counter. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0x Divide the system clock by 1.

Table continues on the next page...

FTMx_DEADTIME field descriptions (continued)

Field	Description
	10 Divide the system clock by 4. 11 Divide the system clock by 16.
DTVAL	Deadtime Value Selects the deadtime insertion value for the deadtime counter. The deadtime counter is clocked by a scaled version of the system clock. See the description of DTPS. Deadtime insert value = (DTPS × DTVAL). DTVAL selects the number of deadtime counts inserted as follows: When DTVAL is 0, no counts are inserted. When DTVAL is 1, 1 count is inserted. When DTVAL is 2, 2 counts are inserted. This pattern continues up to a possible 63 counts. This field is write protected. It can be written only when MODE[WPDIS] = 1.

36.3.16 FTM External Trigger (FTMx_EXTTRIG)

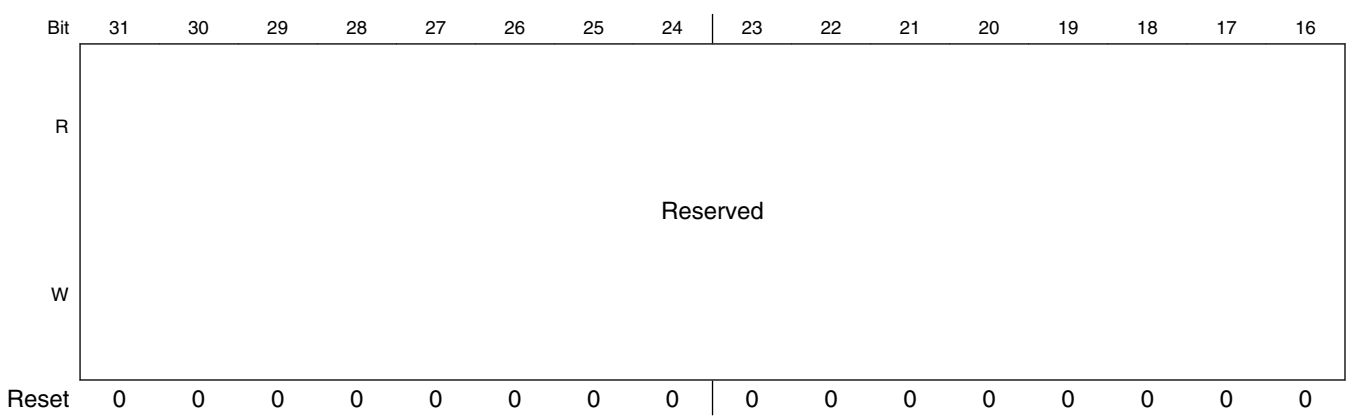
This register:

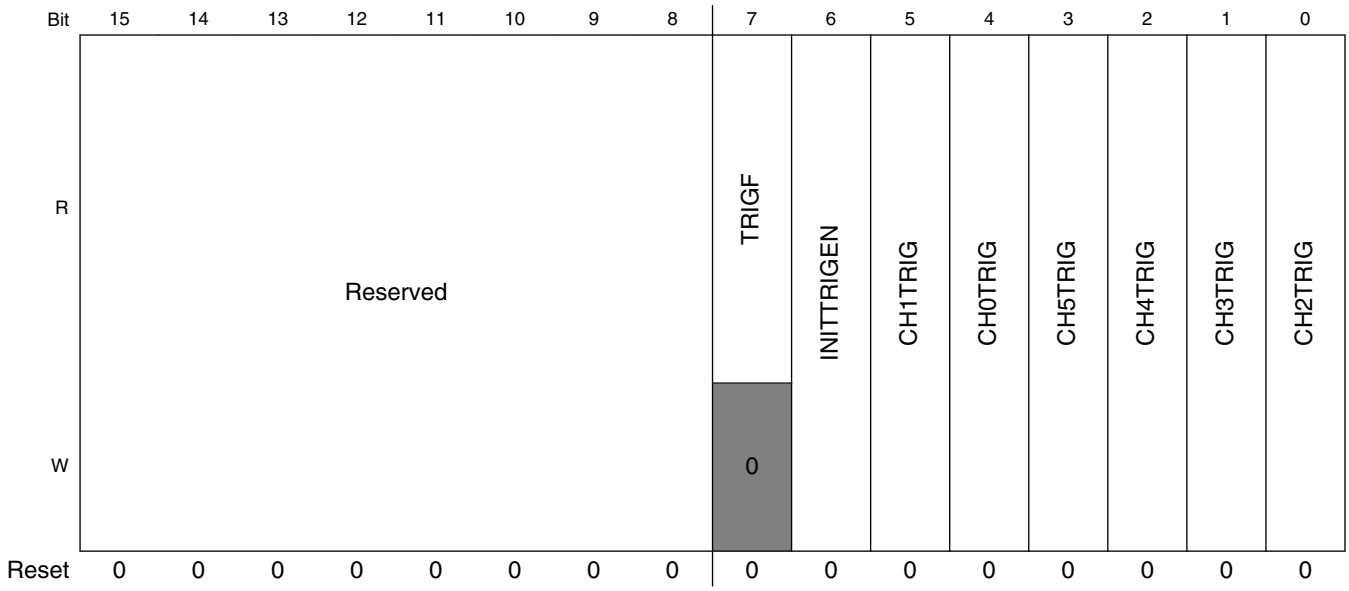
- Indicates when a channel trigger was generated
- Enables the generation of a trigger when the FTM counter is equal to its initial value
- Selects which channels are used in the generation of the channel triggers

Several channels can be selected to generate multiple triggers in one PWM period. See Channel trigger output and Initialization trigger.

Channels 6 and 7 are not used to generate channel triggers.

Address: Base address + 6Ch offset





FTMx_EXTTRIG field descriptions

Field	Description
31–8 Reserved	This field is reserved.
7 TRIGF	Channel Trigger Flag Set by hardware when a channel trigger is generated. Clear TRIGF by reading EXTTRIG while TRIGF is set and then writing a 0 to TRIGF. Writing a 1 to TRIGF has no effect. If another channel trigger is generated before the clearing sequence is completed, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF. 0 No channel trigger was generated. 1 A channel trigger was generated.
6 INITTRIGEN	Initialization Trigger Enable Enables the generation of the trigger when the FTM counter is equal to the CNTIN register. 0 The generation of initialization trigger is disabled. 1 The generation of initialization trigger is enabled.
5 CH1TRIG	Channel 1 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
4 CH0TRIG	Channel 0 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
3 CH5TRIG	Channel 5 Trigger Enable

Table continues on the next page...

FTMx_EXTTRIG field descriptions (continued)

Field	Description
	Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
2 CH4TRIG	Channel 4 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
1 CH3TRIG	Channel 3 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
0 CH2TRIG	Channel 2 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.

36.3.17 Channels Polarity (FTMx_POL)

This register defines the output polarity of the FTM channels.

NOTE

The safe value that is driven in a channel output when the fault control is enabled and a fault condition is detected is the inactive state of the channel. That is, the safe value of a channel is the value of its POL bit.

Address: Base address + 70h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
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 W	Reserved								POL7	POL6	POL5	POL4	POL3	POL2	POL1	POL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_POL field descriptions

Field	Description
31–8 Reserved	This field is reserved.
7 POL7	Channel 7 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
6 POL6	Channel 6 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
5 POL5	Channel 5 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
4 POL4	Channel 4 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
3 POL3	Channel 3 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
2 POL2	Channel 2 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
1 POL1	Channel 1 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

FTMx_POL field descriptions (continued)

Field	Description
	0 The channel polarity is active high. 1 The channel polarity is active low.
0 POL0	Channel 0 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.

36.3.18 Fault Mode Status (FTMx_FMS)

This register contains the fault detection flags, write protection enable bit, and the logic OR of the enabled fault inputs.

Address: Base address + 74h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								FAULTF	WPEN	FAULTIN	0	FAULTF3	FAULTF2	FAULTF1	FAULTF0
W									0				0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_FMS field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 FAULTF	Fault Detection Flag Represents the logic OR of the individual FAULTFj bits where j = 3, 2, 1, 0. Clear FAULTF by reading the FMS register while FAULTF is set and then writing a 0 to FAULTF while there is no existing fault condition at the enabled fault inputs. Writing a 1 to FAULTF has no effect. If another fault condition is detected in an enabled fault input before the clearing sequence is completed, the sequence is reset so FAULTF remains set after the clearing sequence is completed for the earlier fault condition. FAULTF is also cleared when FAULTFj bits are cleared individually. 0 No fault condition was detected. 1 A fault condition was detected.
6 WPEN	Write Protection Enable The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect. 0 Write protection is disabled. Write protected bits can be written. 1 Write protection is enabled. Write protected bits cannot be written.
5 FAULTIN	Fault Inputs Represents the logic OR of the enabled fault inputs after their filter (if their filter is enabled) when fault control is enabled. 0 The logic OR of the enabled fault inputs is 0. 1 The logic OR of the enabled fault inputs is 1.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FAULTF3	Fault Detection Flag 3 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF3 by reading the FMS register while FAULTF3 is set and then writing a 0 to FAULTF3 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF3 has no effect. FAULTF3 bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF3 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.
2 FAULTF2	Fault Detection Flag 2 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF2 by reading the FMS register while FAULTF2 is set and then writing a 0 to FAULTF2 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF2 has no effect. FAULTF2 bit is also cleared when FAULTF bit is cleared.

Table continues on the next page...

FTMx_FMS field descriptions (continued)

Field	Description
	If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF2 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.
1 FAULTF1	Fault Detection Flag 1 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF1 by reading the FMS register while FAULTF1 is set and then writing a 0 to FAULTF1 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF1 has no effect. FAULTF1 bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF1 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.
0 FAULTF0	Fault Detection Flag 0 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF0 by reading the FMS register while FAULTF0 is set and then writing a 0 to FAULTF0 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF0 has no effect. FAULTF0 bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF0 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.

36.3.19 Input Capture Filter Control (FTMx_FILTER)

This register selects the filter value for the inputs of channels.

Channels 4, 5, 6 and 7 do not have an input filter.

NOTE

Writing to the FILTER register has immediate effect and must be done only when the channels 0, 1, 2, and 3 are not in input modes. Failure to do this could result in a missing valid signal.

Address: Base address + 78h offset

Bit	31–16	15–12	11–8	7–4	3–0
R W	Reserved	CH3FVAL	CH2FVAL	CH1FVAL	CH0FVAL
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

FTMx_FILTER field descriptions

Field	Description
31–16 Reserved	This field is reserved.
15–12 CH3FVAL	Channel 3 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
11–8 CH2FVAL	Channel 2 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
7–4 CH1FVAL	Channel 1 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
CH0FVAL	Channel 0 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.

36.3.20 Fault Control (FTMx_FLTCTRL)

This register selects the filter value for the fault inputs, enables the fault inputs and the fault inputs filter.

Address: Base address + 7Ch offset

Bit	31–24	23–16
R	0	
W		
Reset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

Bit	15–12	11–8	7	6	5	4	3	2	1	0
R	0	FFVAL	FFLTR3EN	FFLTR2EN	FFLTR1EN	FFLTR0EN	FAULT3EN	FAULT2EN	FAULT1EN	FAULT0EN
W										
Reset	0 0 0 0	0 0 0 0	0	0	0	0	0	0	0	0

FTMx_FLTCTRL field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 FFVAL	Fault Input Filter Selects the filter value for the fault inputs. The fault filter is disabled when the value is zero. **NOTE:** Writing to this field has immediate effect and must be done only when the fault control or all fault inputs are disabled. Failure to do this could result in a missing fault detection.
7 FFLTR3EN	Fault Input 3 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
6 FFLTR2EN	Fault Input 2 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
5 FFLTR1EN	Fault Input 1 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
4 FFLTR0EN	Fault Input 0 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
3 FAULT3EN	Fault Input 3 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.
2 FAULT2EN	Fault Input 2 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.

Table continues on the next page...

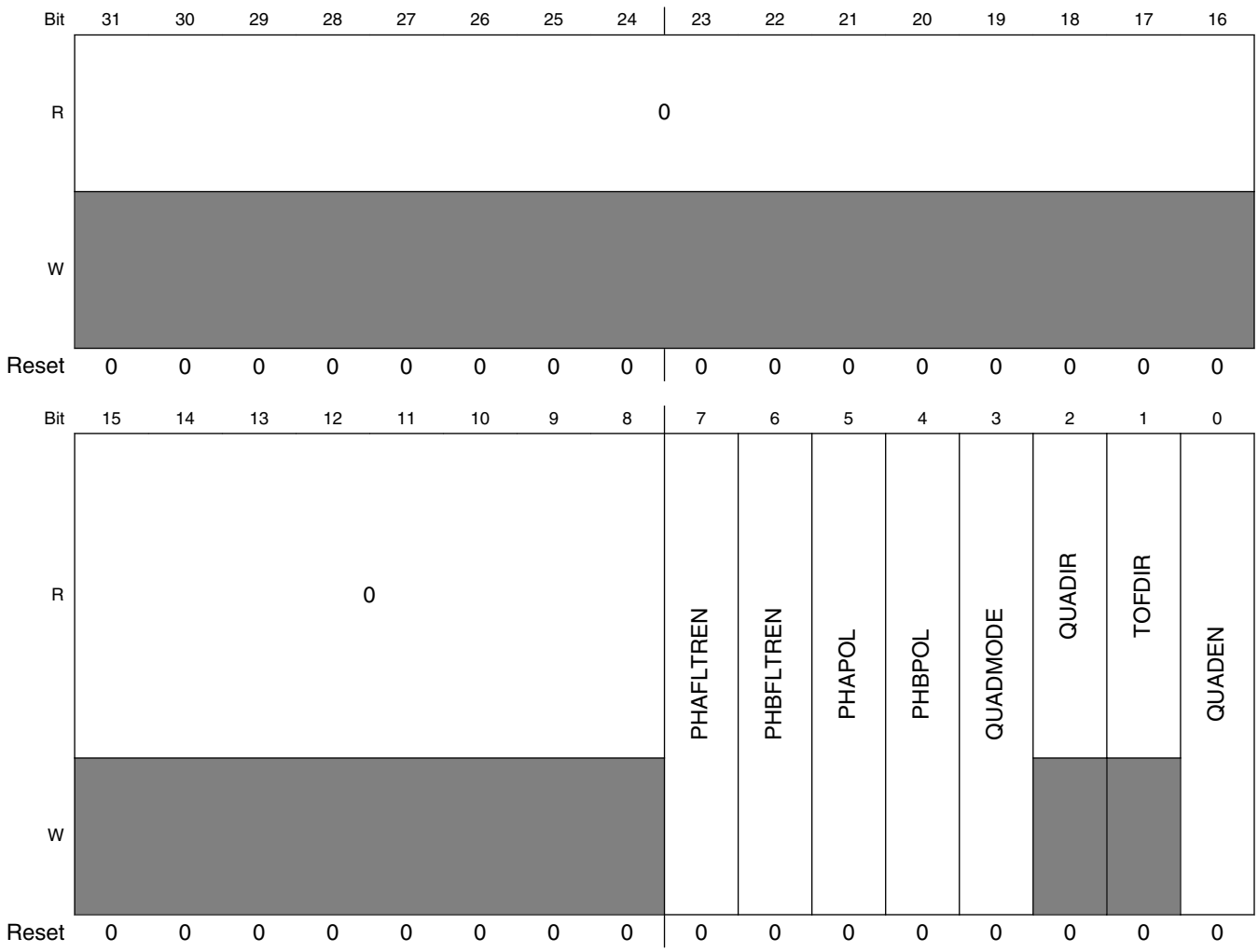
FTMx_FLTCTRL field descriptions (continued)

Field	Description
1 FAULT1EN	Fault Input 1 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.
0 FAULT0EN	Fault Input 0 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.

36.3.21 Quadrature Decoder Control And Status (FTMx_QDCTRL)

This register has the control and status bits for the Quadrature Decoder mode.

Address: Base address + 80h offset



FTMx_QDCTRL field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PHAFLTREN	Phase A Input Filter Enable Enables the filter for the quadrature decoder phase A input. The filter value for the phase A input is defined by the CH0FVAL field of FILTER. The phase A filter is also disabled when CH0FVAL is zero. 0 Phase A input filter is disabled. 1 Phase A input filter is enabled.

Table continues on the next page...

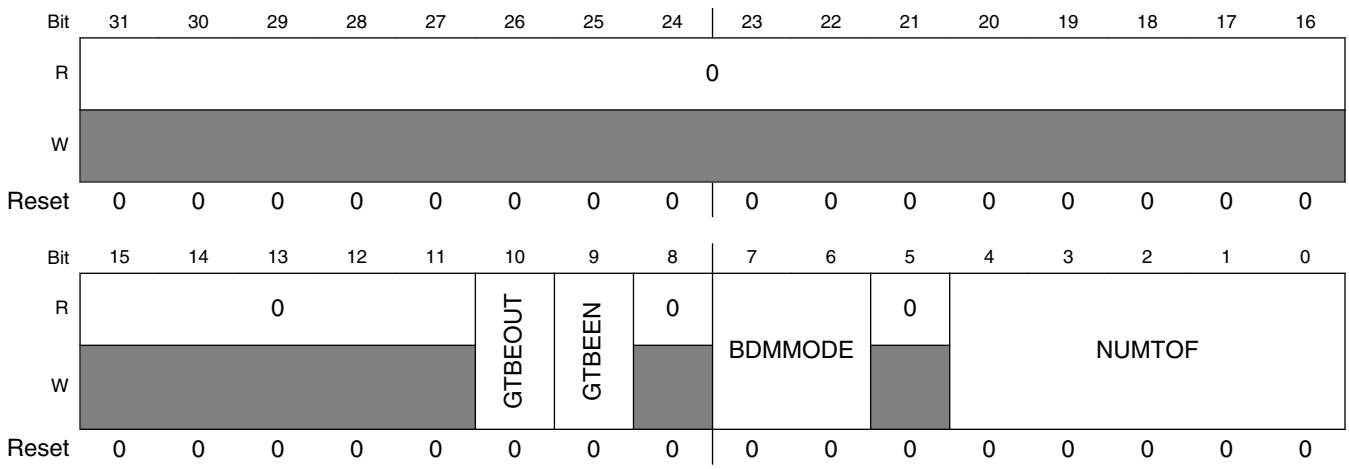
FTMx_QDCTRL field descriptions (continued)

Field	Description
6 PHBFLTREN	Phase B Input Filter Enable Enables the filter for the quadrature decoder phase B input. The filter value for the phase B input is defined by the CH1FVAL field of FILTER. The phase B filter is also disabled when CH1FVAL is zero. 0 Phase B input filter is disabled. 1 Phase B input filter is enabled.
5 PHAPOL	Phase A Input Polarity Selects the polarity for the quadrature decoder phase A input. 0 Normal polarity. Phase A input signal is not inverted before identifying the rising and falling edges of this signal. 1 Inverted polarity. Phase A input signal is inverted before identifying the rising and falling edges of this signal.
4 PHBPOL	Phase B Input Polarity Selects the polarity for the quadrature decoder phase B input. 0 Normal polarity. Phase B input signal is not inverted before identifying the rising and falling edges of this signal. 1 Inverted polarity. Phase B input signal is inverted before identifying the rising and falling edges of this signal.
3 QUADMODE	Quadrature Decoder Mode Selects the encoding mode used in the Quadrature Decoder mode. 0 Phase A and phase B encoding mode. 1 Count and direction encoding mode.
2 QUADIR	FTM Counter Direction In Quadrature Decoder Mode Indicates the counting direction. 0 Counting direction is decreasing (FTM counter decrement). 1 Counting direction is increasing (FTM counter increment).
1 TOFDIR	Timer Overflow Direction In Quadrature Decoder Mode Indicates if the TOF bit was set on the top or the bottom of counting. 0 TOF bit was set on the bottom of counting. There was an FTM counter decrement and FTM counter changes from its minimum value (CNTIN register) to its maximum value (MOD register). 1 TOF bit was set on the top of counting. There was an FTM counter increment and FTM counter changes from its maximum value (MOD register) to its minimum value (CNTIN register).
0 QUADEN	Quadrature Decoder Mode Enable Enables the Quadrature Decoder mode. In this mode, the phase A and B input signals control the FTM counter direction. The Quadrature Decoder mode has precedence over the other modes. See Table 36-2. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Quadrature Decoder mode is disabled. 1 Quadrature Decoder mode is enabled.

36.3.22 Configuration (FTMx_CONF)

This register selects the number of times that the FTM counter overflow should occur before the TOF bit to be set, the FTM behavior in BDM modes, the use of an external global time base, and the global time base signal generation.

Address: Base address + 84h offset



FTMx_CONF field descriptions

Field	Description
31–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 GTBEOUT	Global Time Base Output Enables the global time base signal generation to other FTMs. 0 A global time base signal generation is disabled. 1 A global time base signal generation is enabled.
9 GTBEEN	Global Time Base Enable Configures the FTM to use an external global time base signal that is generated by another FTM. 0 Use of an external global time base is disabled. 1 Use of an external global time base is enabled.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 BDMMODE	BDM Mode Selects the FTM behavior in BDM mode. See BDM mode.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
NUMTOF	TOF Frequency

Table continues on the next page...

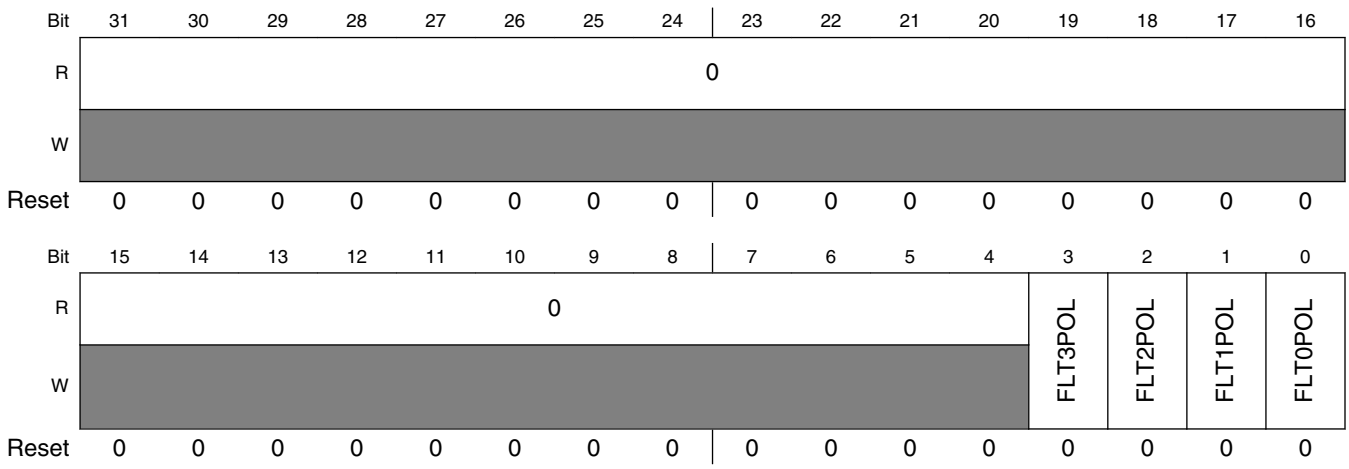
FTMx_CONF field descriptions (continued)

Field	Description
	Selects the ratio between the number of counter overflows to the number of times the TOF bit is set. NUMTOF = 0: The TOF bit is set for each counter overflow. NUMTOF = 1: The TOF bit is set for the first counter overflow but not for the next overflow. NUMTOF = 2: The TOF bit is set for the first counter overflow but not for the next 2 overflows. NUMTOF = 3: The TOF bit is set for the first counter overflow but not for the next 3 overflows. This pattern continues up to a maximum of 31.

36.3.23 FTM Fault Input Polarity (FTMx_FLTPOL)

This register defines the fault inputs polarity.

Address: Base address + 88h offset



FTMx_FLTPOL field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FLT3POL	Fault Input 3 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.
2 FLT2POL	Fault Input 2 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

FTMx_FLTPOL field descriptions (continued)

Field	Description
	0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.
1 FLT1POL	Fault Input 1 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.
0 FLT0POL	Fault Input 0 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.

36.3.24 Synchronization Configuration (FTMx_SYNCONF)

This register selects the PWM synchronization configuration, SWOCTRL, INVCTRL and CNTIN registers synchronization, if FTM clears the TRIGj bit, where j = 0, 1, 2, when the hardware trigger j is detected.

Address: Base address + 8Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0											HWSOC	HWINVC	HWOM	HWWRBUF	HWRSTCNT
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			SWSOC	SWINVC	SWOM	SWWRBUF	SWRSTCNT	SYNCMODE	0	SWOC	INVC	0	CNTINC	0	HWTRIGMODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SYNCONF field descriptions

Field	Description
31–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FTMx_SYNCONF field descriptions (continued)

Field	Description
20 HWSOC	Software output control synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the SWOCTRL register synchronization. 1 A hardware trigger activates the SWOCTRL register synchronization.
19 HWINVC	Inverting control synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the INVCTRL register synchronization. 1 A hardware trigger activates the INVCTRL register synchronization.
18 HWOM	Output mask synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the OUTMASK register synchronization. 1 A hardware trigger activates the OUTMASK register synchronization.
17 HWWRBUF	MOD, CNTIN, and CV registers synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate MOD, CNTIN, and CV registers synchronization. 1 A hardware trigger activates MOD, CNTIN, and CV registers synchronization.
16 HWRSTCNT	FTM counter synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the FTM counter synchronization. 1 A hardware trigger activates the FTM counter synchronization.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 SWSOC	Software output control synchronization is activated by the software trigger. 0 The software trigger does not activate the SWOCTRL register synchronization. 1 The software trigger activates the SWOCTRL register synchronization.
11 SWINVC	Inverting control synchronization is activated by the software trigger. 0 The software trigger does not activate the INVCTRL register synchronization. 1 The software trigger activates the INVCTRL register synchronization.
10 SWOM	Output mask synchronization is activated by the software trigger. 0 The software trigger does not activate the OUTMASK register synchronization. 1 The software trigger activates the OUTMASK register synchronization.
9 SWWRBUF	MOD, CNTIN, and CV registers synchronization is activated by the software trigger. 0 The software trigger does not activate MOD, CNTIN, and CV registers synchronization. 1 The software trigger activates MOD, CNTIN, and CV registers synchronization.
8 SWRSTCNT	FTM counter synchronization is activated by the software trigger. 0 The software trigger does not activate the FTM counter synchronization. 1 The software trigger activates the FTM counter synchronization.
7 SYNCMODE	Synchronization Mode Selects the PWM Synchronization mode. 0 Legacy PWM synchronization is selected. 1 Enhanced PWM synchronization is selected.

Table continues on the next page...

FTMx_SYNCONF field descriptions (continued)

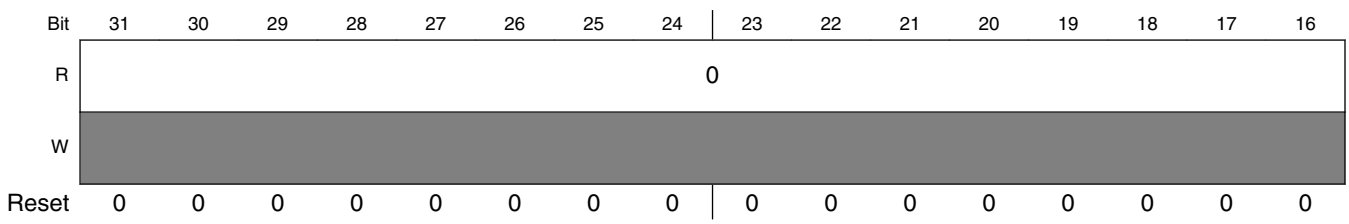
Field	Description
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SWOC	SWOCTRL Register Synchronization 0 SWOCTRL register is updated with its buffer value at all rising edges of system clock. 1 SWOCTRL register is updated with its buffer value by the PWM synchronization.
4 INVC	INVCTRL Register Synchronization 0 INVCTRL register is updated with its buffer value at all rising edges of system clock. 1 INVCTRL register is updated with its buffer value by the PWM synchronization.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CNTINC	CNTIN Register Synchronization 0 CNTIN register is updated with its buffer value at all rising edges of system clock. 1 CNTIN register is updated with its buffer value by the PWM synchronization.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 HWTRIGMODE	Hardware Trigger Mode 0 FTM clears the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2. 1 FTM does not clear the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2.

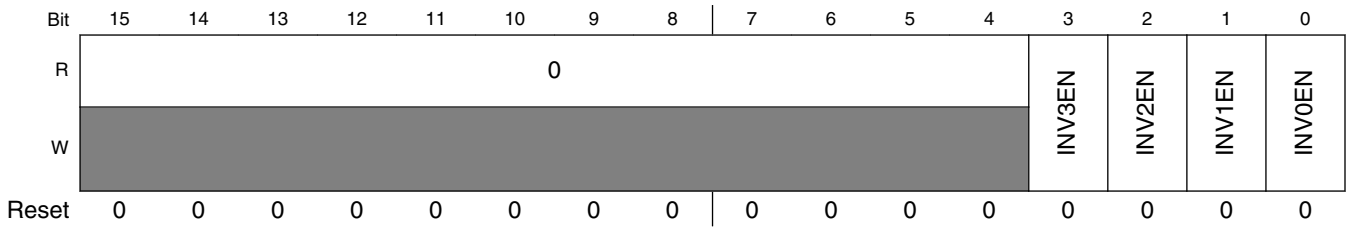
36.3.25 FTM Inverting Control (FTMx_INVCTRL)

This register controls when the channel (n) output becomes the channel (n+1) output, and channel (n+1) output becomes the channel (n) output. Each INVmEN bit enables the inverting operation for the corresponding pair channels m.

This register has a write buffer. The INVmEN bit is updated by the INVCTRL register synchronization.

Address: Base address + 90h offset





FTMx_INVCTRL field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 INV3EN	Pair Channels 3 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
2 INV2EN	Pair Channels 2 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
1 INV1EN	Pair Channels 1 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
0 INV0EN	Pair Channels 0 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.

36.3.26 FTM Software Output Control (FTMx_SWOCTRL)

This register enables software control of channel (n) output and defines the value forced to the channel (n) output:

- The CHnOC bits enable the control of the corresponding channel (n) output by software.
- The CHnOCV bits select the value that is forced at the corresponding channel (n) output.

This register has a write buffer. The fields are updated by the SWOCTRL register synchronization.

Address: Base address + 94h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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	CH7OCV	CH6OCV	CH5OCV	CH4OCV	CH3OCV	CH2OCV	CH1OCV	CH0OCV	CH7OC	CH6OC	CH5OC	CH4OC	CH3OC	CH2OC	CH1OC	CH0OC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SWOCTRL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CH7OCV	Channel 7 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
14 CH6OCV	Channel 6 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
13 CH5OCV	Channel 5 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
12 CH4OCV	Channel 4 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
11 CH3OCV	Channel 3 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
10 CH2OCV	Channel 2 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
9 CH1OCV	Channel 1 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
8 CH0OCV	Channel 0 Software Output Control Value

Table continues on the next page...

FTMx_SWOCTRL field descriptions (continued)

Field	Description
	0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
7 CH7OC	Channel 7 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
6 CH6OC	Channel 6 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
5 CH5OC	Channel 5 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
4 CH4OC	Channel 4 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
3 CH3OC	Channel 3 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
2 CH2OC	Channel 2 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
1 CH1OC	Channel 1 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
0 CH0OC	Channel 0 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.

36.3.27 FTM PWM Load (FTMx_PWMLOAD)

Enables the loading of the MOD, CNTIN, C(n)V, and C(n+1)V registers with the values of their write buffers when the FTM counter changes from the MOD register value to its next value or when a channel (j) match occurs. A match occurs for the channel (j) when FTM counter = C(j)V.

Address: Base address + 98h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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						LDOK	0	CH7SEL	CH6SEL	CH5SEL	CH4SEL	CH3SEL	CH2SEL	CH1SEL	CH0SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_PWMLOAD field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 LDOK	Load Enable Enables the loading of the MOD, CNTIN, and CV registers with the values of their write buffers. 0 Loading updated values is disabled. 1 Loading updated values is enabled.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CH7SEL	Channel 7 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
6 CH6SEL	Channel 6 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
5 CH5SEL	Channel 5 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.

Table continues on the next page...

FTMx_PWMLOAD field descriptions (continued)

Field	Description
4 CH4SEL	Channel 4 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
3 CH3SEL	Channel 3 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
2 CH2SEL	Channel 2 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
1 CH1SEL	Channel 1 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
0 CH0SEL	Channel 0 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.

36.4 Functional description

The notation used in this document to represent the counters and the generation of the signals is shown in the following figure.

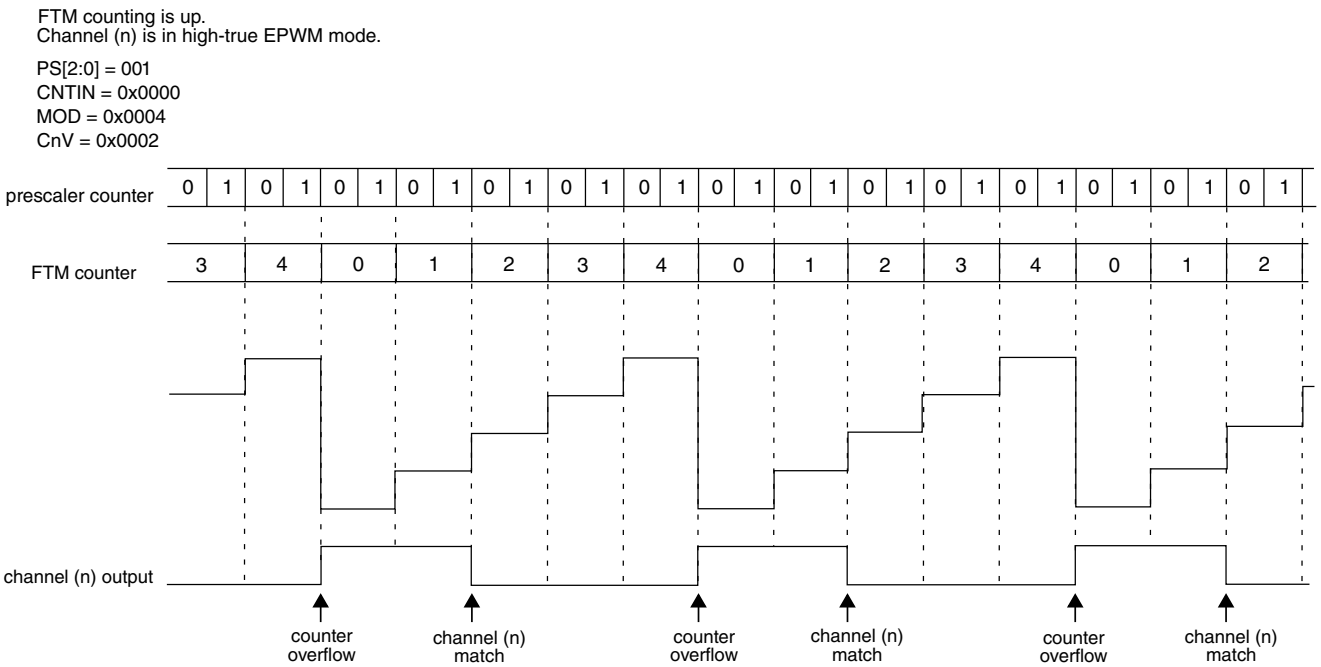


Figure 36-2. Notation used

36.4.1 Clock source

The FTM has only one clock domain: the system clock.

36.4.1.1 Counter clock source

The CLKS[1:0] bits in the SC register selects clock sources for the FTM counter or disables the FTM counter. After any chip reset, CLKS[1:0] = 0:0 so no clock source is selected.

The CLKS[1:0] bits may be read or written at any time. Disabling the FTM counter by writing 0:0 to the CLKS[1:0] bits does not affect the FTM counter value or other registers.

The fixed frequency clock is an alternative clock source for the FTM counter that allows the selection of a clock other than the system clock or an external clock. This clock input is defined by chip integration; see the chip-specific FTM information for further details. Due to FTM hardware implementation limitations, the frequency of the fixed frequency clock must not exceed 1/2 of the system clock frequency.

The external clock passes through a synchronizer clocked by the system clock to assure that counter transitions are properly aligned to system clock transitions.Therefore, to meet Nyquist criteria considering also jitter, the frequency of the external clock source must not exceed 1/4 of the system clock frequency.

36.4.2 Prescaler

The selected counter clock source passes through a prescaler that is a 7-bit counter. The value of the prescaler is selected by the PS[2:0] bits. The following figure shows an example of the prescaler counter and FTM counter.

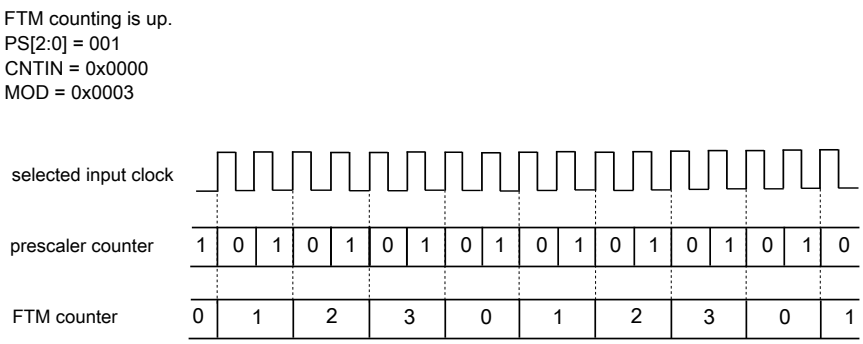


Figure 36-3. Example of the prescaler counter

36.4.3 Counter

The FTM has a 16-bit counter that is used by the channels either for input or output modes. The FTM counter clock is the selected clock divided by the prescaler.

The FTM counter has these modes of operation:

- Up counting
- Up-down counting
- Quadrature Decoder mode

36.4.3.1 Up counting

Up counting is selected when:

- QUADEN = 0, and
- CPWMS = 0

CNTIN defines the starting value of the count and MOD defines the final value of the count, see the following figure. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is reloaded with the value of CNTIN.

The FTM period when using up counting is (MOD – CNTIN + 0x0001) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to CNTIN.

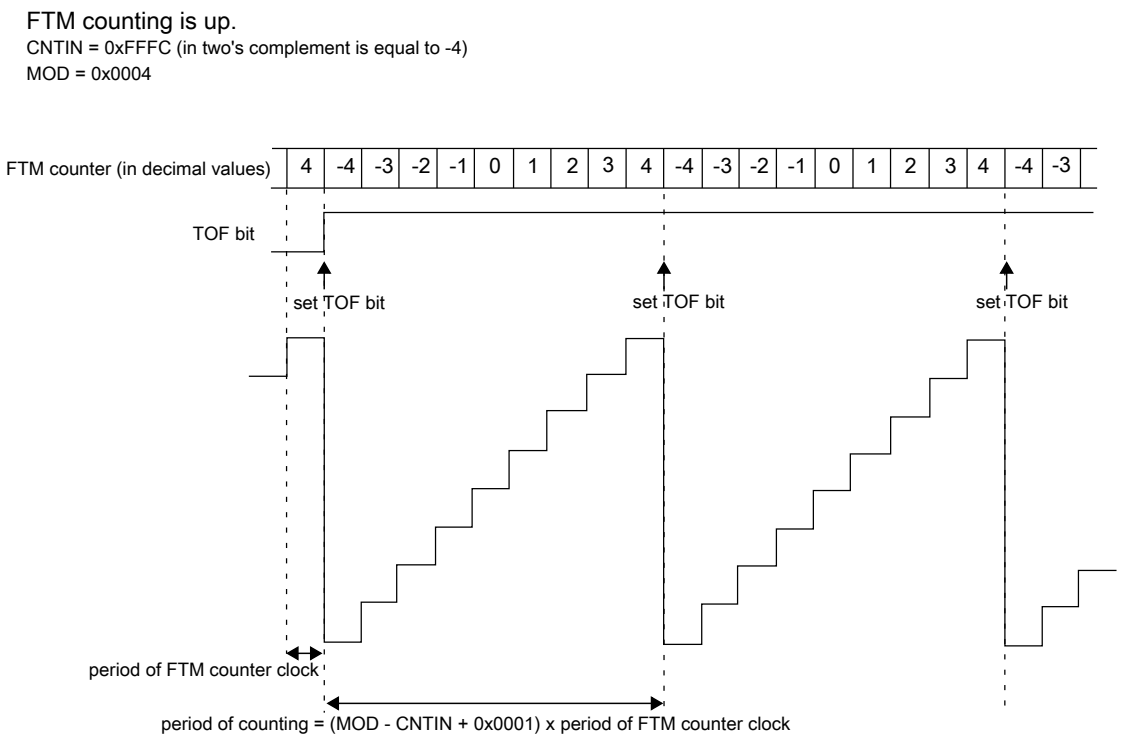


Figure 36-4. Example of FTM up and signed counting

Table 36-4. FTM counting based on CNTIN value

When	Then
CNTIN = 0x0000	The FTM counting is equivalent to TPM up counting, that is, up and unsigned counting. See the following figure.
CNTIN[15] = 1	The initial value of the FTM counter is a negative number in two's complement, so the FTM counting is up and signed.
CNTIN[15] = 0 and CNTIN ≠ 0x0000	The initial value of the FTM counter is a positive number, so the FTM counting is up and unsigned.

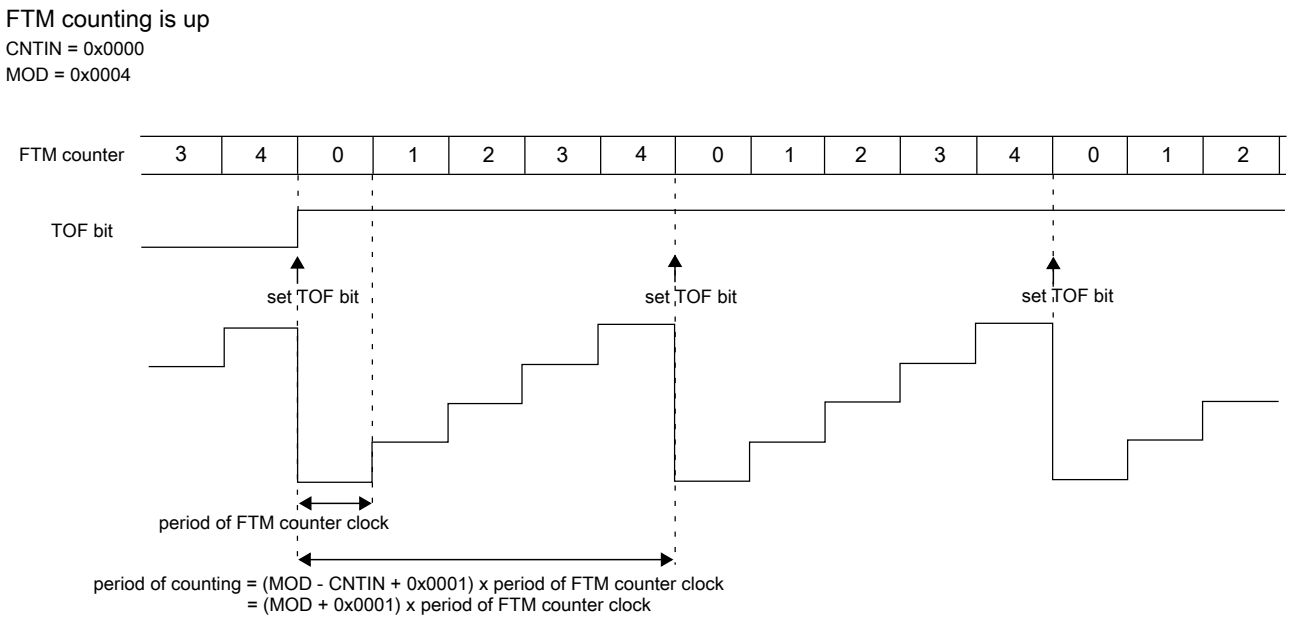


Figure 36-5. Example of FTM up counting with CNTIN = 0x0000

Note

- FTM operation is only valid when the value of the CNTIN register is less than the value of the MOD register, either in the unsigned counting or signed counting. It is the responsibility of the software to ensure that the values in the CNTIN and MOD registers meet this requirement. Any values of CNTIN and MOD that do not satisfy this criteria can result in unpredictable behavior.
- MOD = CNTIN is a redundant condition. In this case, the FTM counter is always equal to MOD and the TOF bit is set in each rising edge of the FTM counter clock.
- When MOD = 0x0000, CNTIN = 0x0000, for example after reset, and FTMEN = 1, the FTM counter remains stopped at 0x0000 until a non-zero value is written into the MOD or CNTIN registers.
- Setting CNTIN to be greater than the value of MOD is not recommended as this unusual setting may make the FTM operation difficult to comprehend. However, there is no restriction on this configuration, and an example is shown in the following figure.

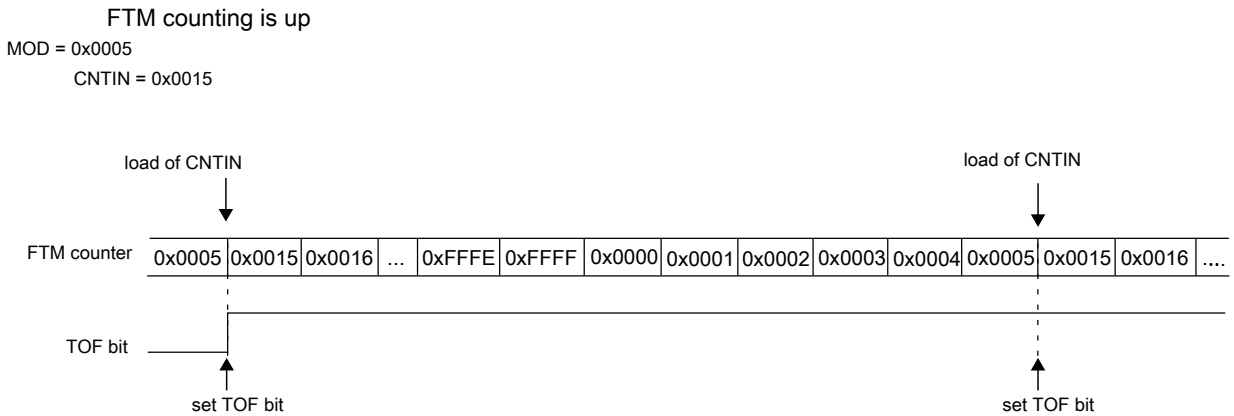


Figure 36-6. Example of up counting when the value of CNTIN is greater than the value of MOD

36.4.3.2 Up-down counting

Up-down counting is selected when:

- QUADEN = 0, and
- CPWMS = 1

CNTIN defines the starting value of the count and MOD defines the final value of the count. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is decremented until it returns to the value of CNTIN and the up-down counting restarts.

The FTM period when using up-down counting is 2 × (MOD – CNTIN) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to (MOD – 1).

If (CNTIN = 0x0000), the FTM counting is equivalent to TPM up-down counting, that is, up-down and unsigned counting. See the following figure.

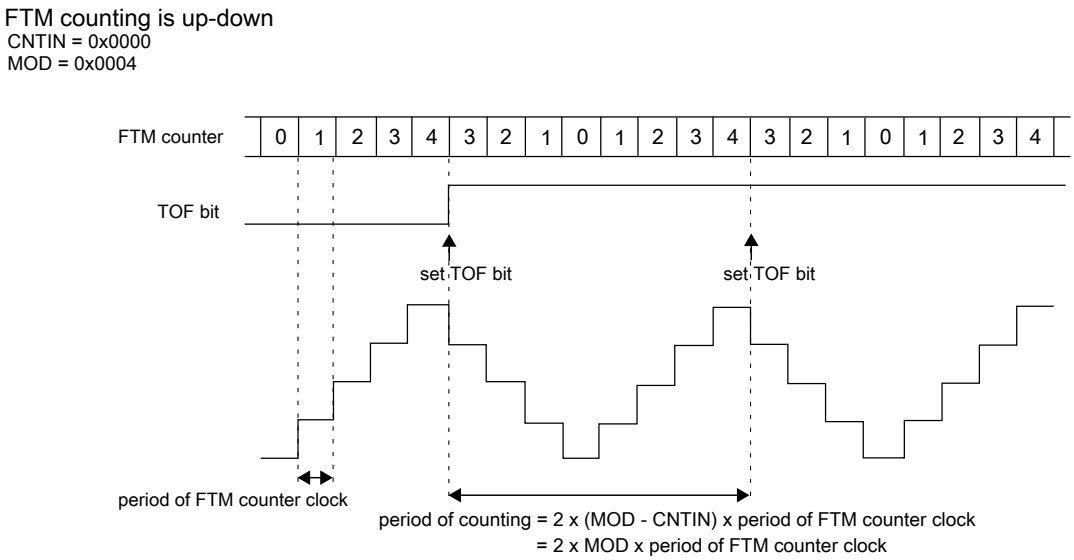


Figure 36-7. Example of up-down counting when CNTIN = 0x0000

Note

When CNTIN is different from zero in the up-down counting, a valid CPWM signal is generated:

- if CnV > CNTIN, or
- if CnV = 0 or if CnV[15] = 1. In this case, 0% CPWM is generated.

36.4.3.3 Free running counter

If (FTMEN = 0) and (MOD = 0x0000 or MOD = 0xFFFF), the FTM counter is a free running counter. In this case, the FTM counter runs free from 0x0000 through 0xFFFF and the TOF bit is set when the FTM counter changes from 0xFFFF to 0x0000. See the following figure.

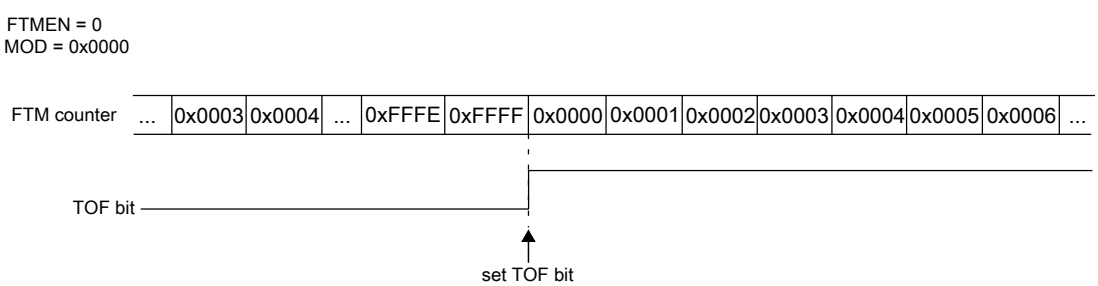


Figure 36-8. Example when the FTM counter is free running

The FTM counter is also a free running counter when:

- FTMEN = 1
- QUADEN = 0
- CPWMS = 0
- CNTIN = 0x0000, and
- MOD = 0xFFFF

36.4.3.4 Counter reset

Any one of the following cases resets the FTM counter to the value in the CNTIN register and the channels output to its initial value, except for channels in Output Compare mode.

- Any write to CNT.
- FTM counter synchronization.
- A channel in Input Capture mode with ICRST = 1 (FTM Counter Reset in Input Capture Mode).

36.4.3.5 When the TOF bit is set

The NUMTOF[4:0] bits define the number of times that the FTM counter overflow should occur before the TOF bit to be set. If NUMTOF[4:0] = 0x00, then the TOF bit is set at each FTM counter overflow.

Initialize the FTM counter, by writing to CNT, after writing to the NUMTOF[4:0] bits to avoid confusion about when the first counter overflow will occur.

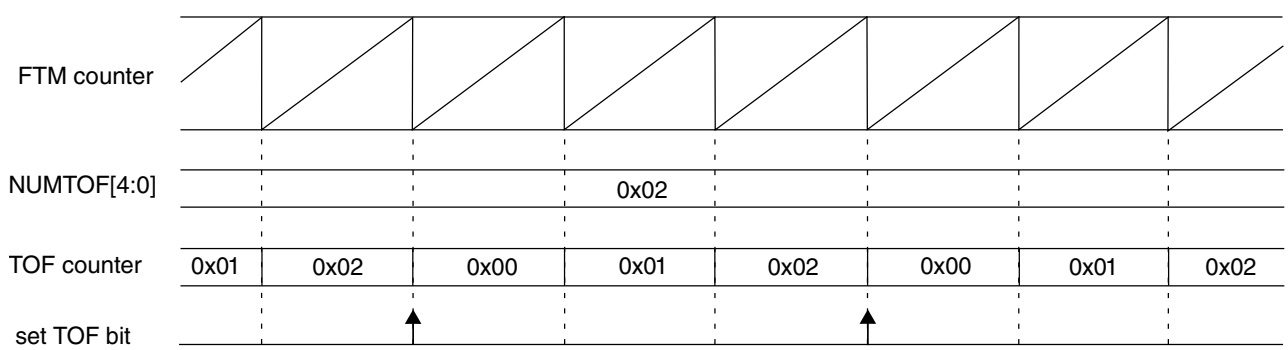


Figure 36-9. Periodic TOF when NUMTOF = 0x02

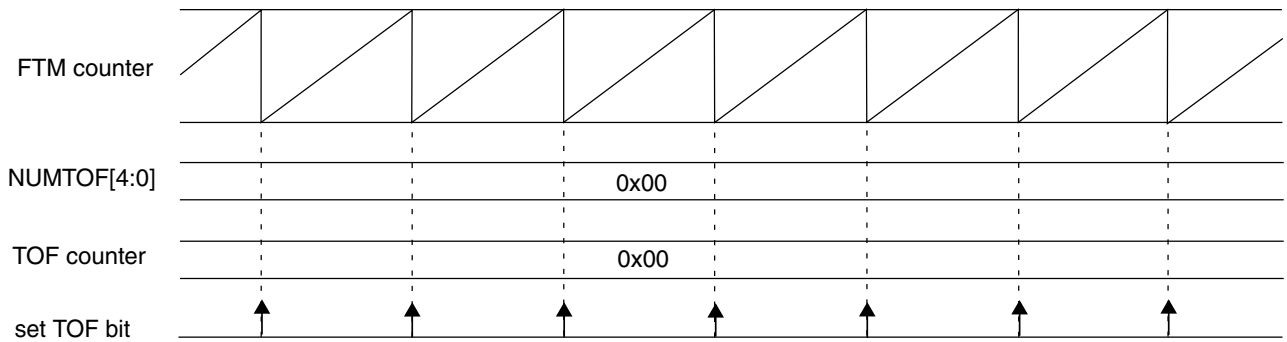


Figure 36-10. Periodic TOF when NUMTOF = 0x00

36.4.4 Input Capture mode

The Input Capture mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0
- MSnB:MSnA = 0:0, and
- ELSnB:ELSnA ≠ 0:0

When a selected edge occurs on the channel input, the current value of the FTM counter is captured into the CnV register, at the same time the CHnF bit is set and the channel interrupt is generated if enabled by CHnIE = 1. See the following figure.

When a channel is configured for input capture, the FTMxCHn pin is an edge-sensitive input. ELSnB:ELSnA control bits determine which edge, falling or rising, triggers input-capture event. Note that the maximum frequency for the channel input signal to be detected correctly is system clock divided by 4, which is required to meet Nyquist criteria for signal sampling.

Writes to the CnV register is ignored in Input Capture mode.

While in BDM, the input capture function works as configured. When a selected edge event occurs, the FTM counter value, which is frozen because of BDM, is captured into the CnV register and the CHnF bit is set.

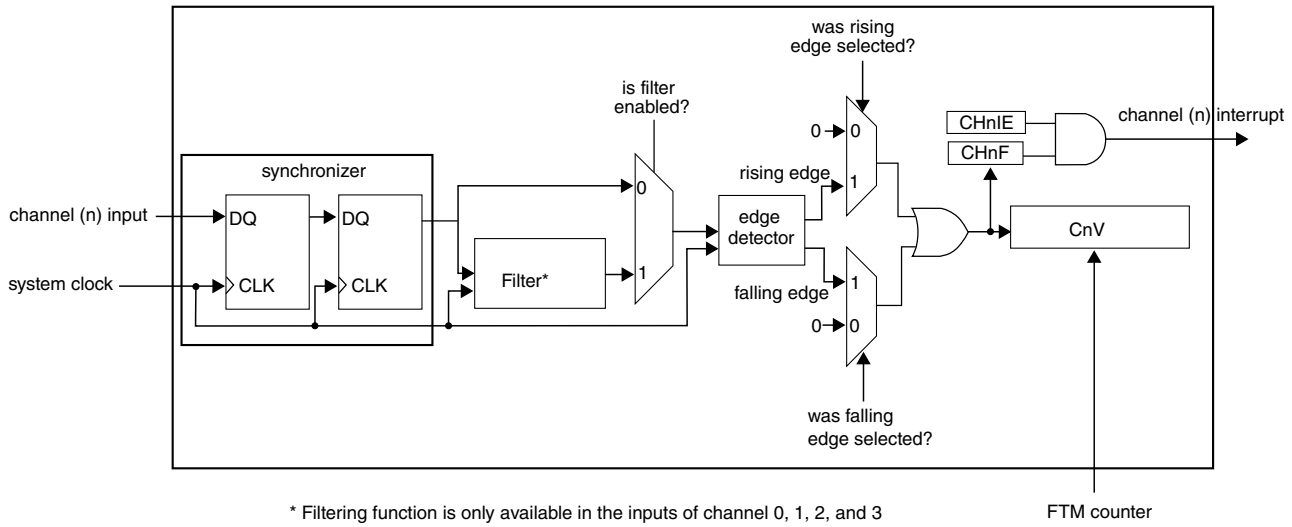


Figure 36-11. Input Capture mode

If the channel input does not have a filter enabled, then the input signal is always delayed 3 rising edges of the system clock, that is, two rising edges to the synchronizer plus one more rising edge to the edge detector. In other words, the CHnF bit is set on the third rising edge of the system clock after a valid edge occurs on the channel input.

36.4.4.1 Filter for Input Capture mode

The filter function is only available on channels 0, 1, 2, and 3.

First, the input signal is synchronized by the system clock. Following synchronization, the input signal enters the filter block. See the following figure.

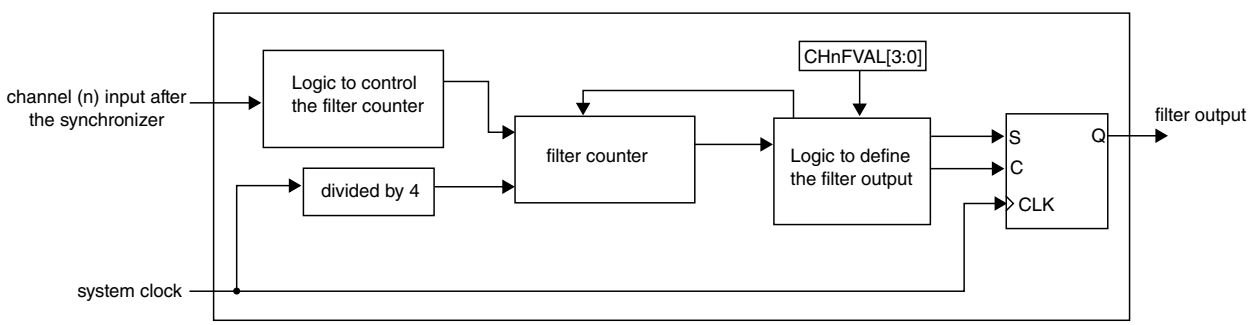


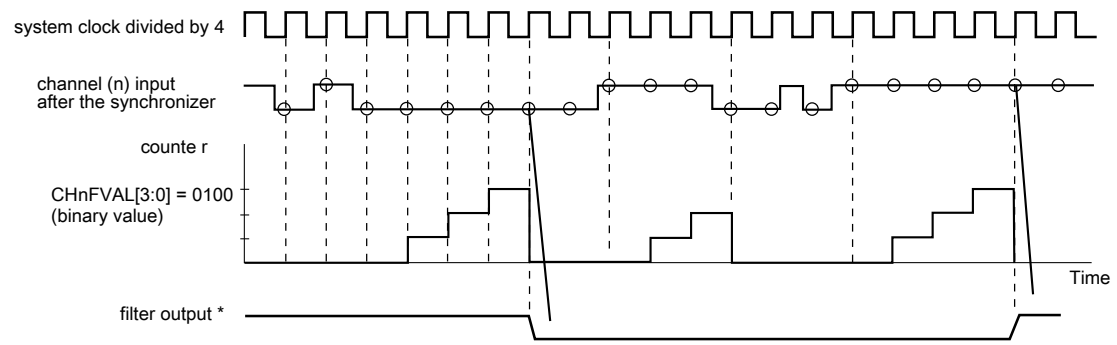
Figure 36-12. Channel input filter

When there is a state change in the input signal, the counter is reset and starts counting up. As long as the new state is stable on the input, the counter continues to increment. When the counter is equal to CHnFVAL[3:0], the state change of the input signal is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the input signal before it can be validated, the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by CHnFVAL[3:0] (× 4 system clocks) is regarded as a glitch and is not passed on to the edge detector. A timing diagram of the input filter is shown in the following figure.

The filter function is disabled when CHnFVAL[3:0] bits are zero. In this case, the input signal is delayed 3 rising edges of the system clock. If (CHnFVAL[3:0] ≠ 0000), then the input signal is delayed by the minimum pulse width (CHnFVAL[3:0] × 4 system clocks) plus a further 4 rising edges of the system clock: two rising edges to the synchronizer, one rising edge to the filter output, plus one more to the edge detector. In other words, CHnF is set (4 + 4 × CHnFVAL[3:0]) system clock periods after a valid edge occurs on the channel input.

The clock for the counter in the channel input filter is the system clock divided by 4.



* Note: Filter output is delayed one system clock of filter counter logic output.

Figure 36-13. Channel input filter example

The figure below shows an example of input capture with filter enabled and the delay added by each part of the input capture logic. Note that the input signal is delayed only by the synchronizer and edge dector logic if the filter is disabled.

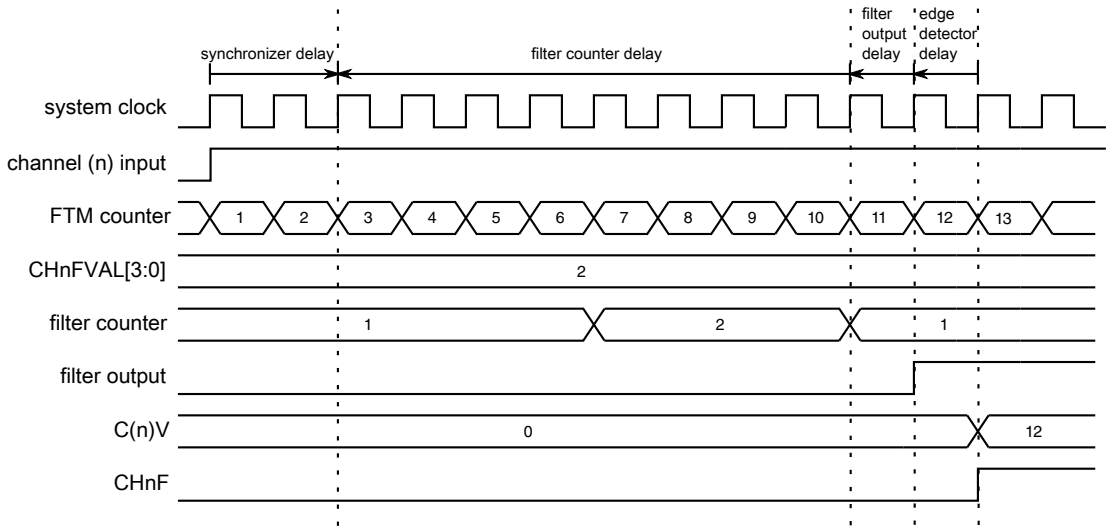


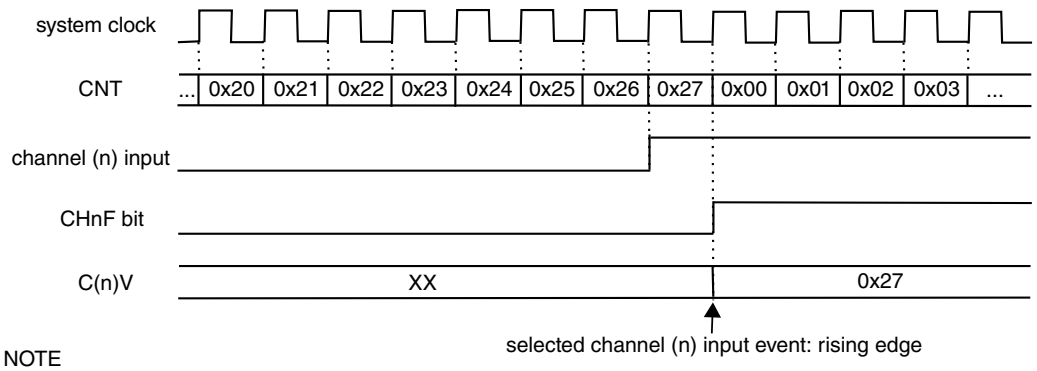
Figure 36-14. Input capture example

36.4.4.2 FTM Counter Reset in Input Capture Mode

If the channel (n) is in input capture mode and FTMx_CnSC [ICRST = 1], then when the selected input capture event occurs in the channel (n) input signal, the current value of the FTM counter is captured into the CnV register, the CHnF bit is set, the channel (n) interrupt is generated (if CHnIE = 1) and the FTM counter is reset to the CNTIN register value.

This allows the FTM to measure a period/pulse being applied to FTM_CHn (counts of the FTM clock input) without having to implement a subtraction calculation in software subsequent to the event occurring.

The figure below shows the FTM counter reset when the selected input capture event is detected in a channel in input capture mode with ICRST = 1.



NOTE
Channel (n) input after its synchronizer and filter
MOD = 0xFFFF
CNTIN = 0x0000
PS[2:0] = 3'b000
ICRST = 1'b1

Figure 36-15. Example of the Input Capture mode with ICRST = 1

NOTE

- It is expected that the ICRST bit be set only when the channel is in input capture mode.
- In this case, if the FTM counter is reset, then the prescaler counter (Prescaler) and the TOF counter (When the TOF bit is set) also are reset.

36.4.5 Output Compare mode

The Output Compare mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB:MSnA = 0:1

In Output Compare mode, the FTM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CnV register of an output compare channel, the channel (n) output can be set, cleared, or toggled.

When a channel is initially configured to Toggle mode, the previous value of the channel output is held until the first output compare event occurs.

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV).

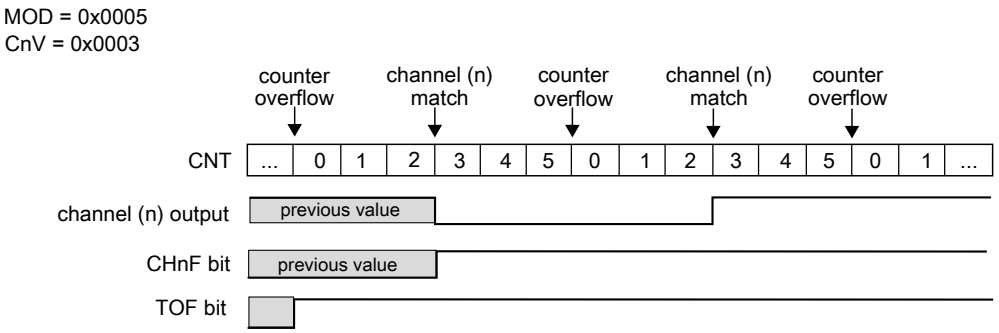


Figure 36-16. Example of the Output Compare mode when the match toggles the channel output

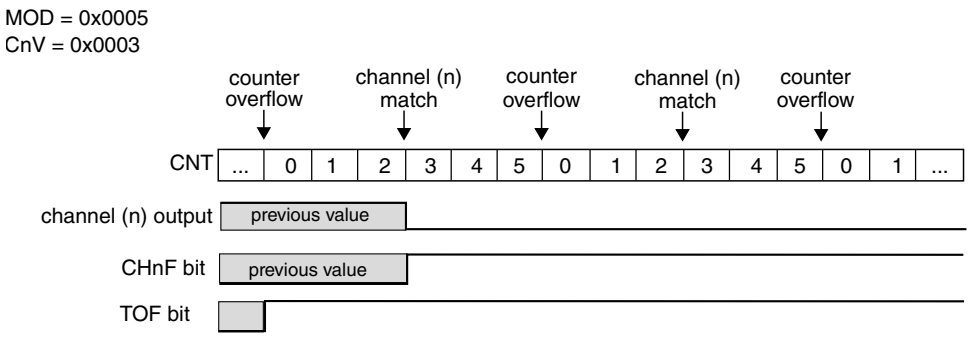


Figure 36-17. Example of the Output Compare mode when the match clears the channel output

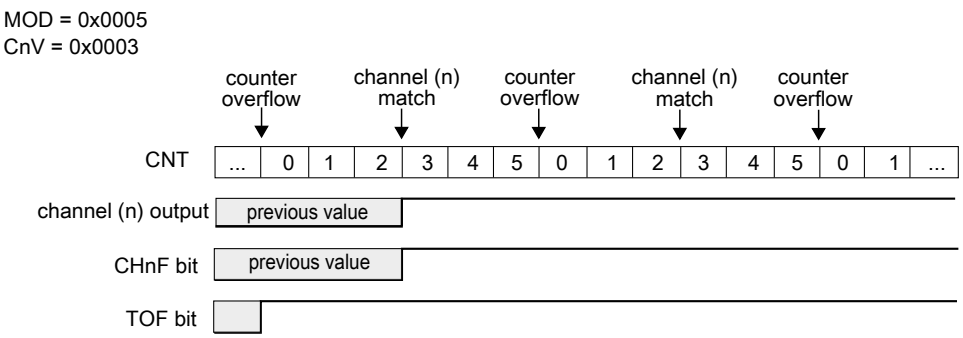


Figure 36-18. Example of the Output Compare mode when the match sets the channel output

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not modified and controlled by FTM.

36.4.6 Edge-Aligned PWM (EPWM) mode

The Edge-Aligned mode is selected when:

- QUADEN = 0

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB = 1

The EPWM period is determined by (MOD − CNTIN + 0x0001) and the pulse width (duty cycle) is determined by (CnV − CNTIN).

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV), that is, at the end of the pulse width.

This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an FTM.

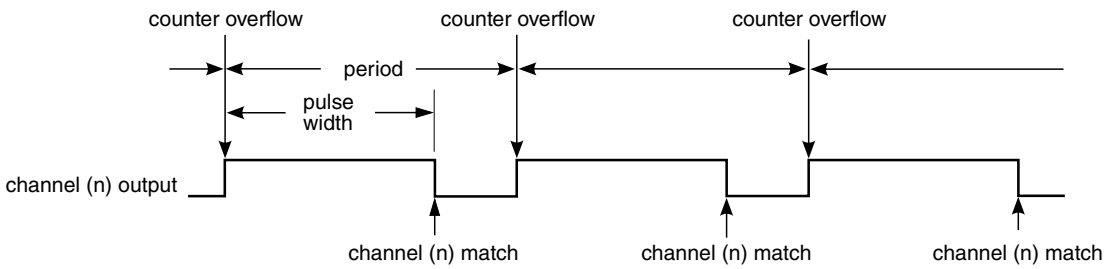


Figure 36-19. EPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced low at the channel (n) match (FTM counter = CnV). See the following figure.

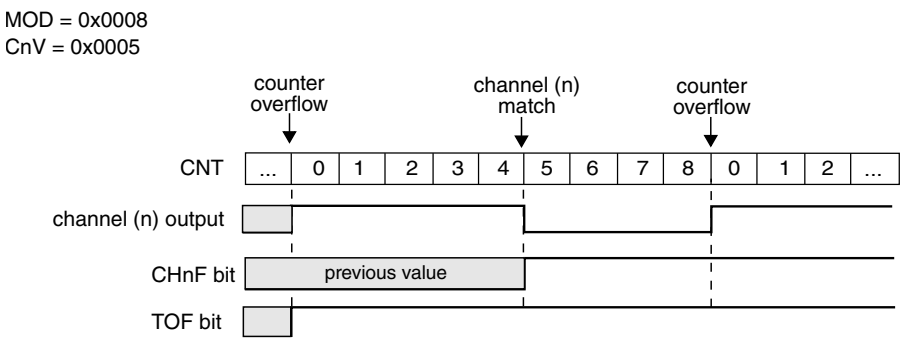


Figure 36-20. EPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced high at the channel (n) match (FTM counter = CnV). See the following figure.

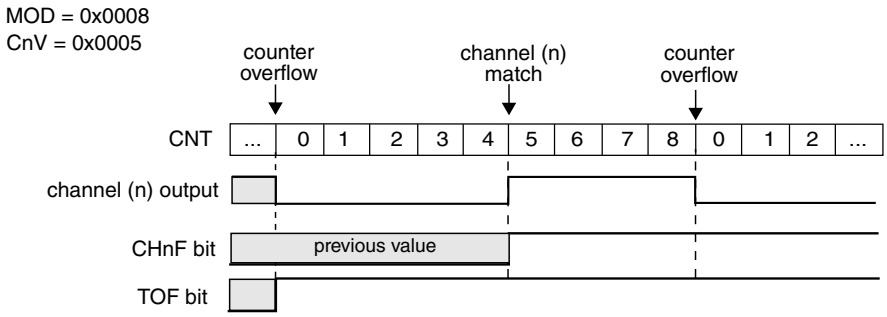


Figure 36-21. EPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000), then the channel (n) output is a 0% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match.

If (CnV > MOD), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set.Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

Note

When CNTIN is different from zero the following EPWM signals can be generated:

- 0% EPWM signal if CnV = CNTIN,
- EPWM signal between 0% and 100% if CNTIN < CnV <= MOD,
- 100% EPWM signal when CNTIN > CnV or CnV > MOD.

36.4.7 Center-Aligned PWM (CPWM) mode

The Center-Aligned mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMBINE = 0, and
- CPWMS = 1

The CPWM pulse width (duty cycle) is determined by 2 × (CnV − CNTIN) and the period is determined by 2 × (MOD − CNTIN). See the following figure. MOD must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

In the CPWM mode, the FTM counter counts up until it reaches MOD and then counts down until it reaches CNTIN.

The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = CnV) when the FTM counting is down (at the begin of the pulse width) and when the FTM counting is up (at the end of the pulse width).

This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value of CNTIN.

The other channel modes are not compatible with the up-down counter (CPWMS = 1). Therefore, all FTM channels must be used in CPWM mode when (CPWMS = 1).

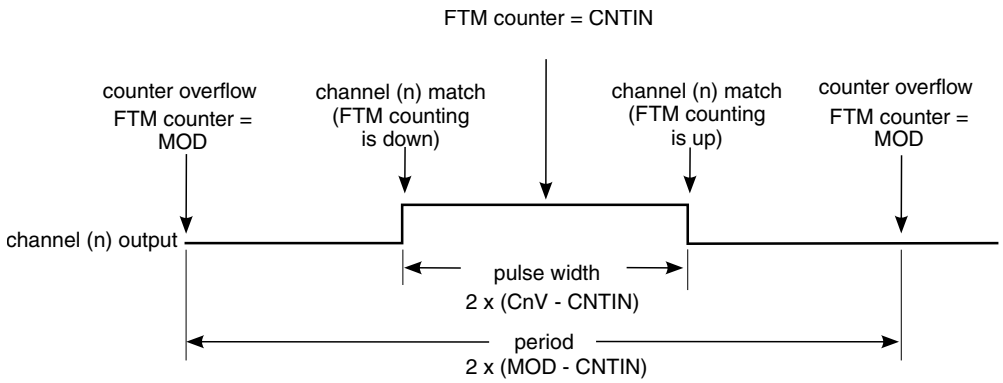


Figure 36-22. CPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the FTM counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the channel (n) match (FTM counter = CnV) when counting down, and it is forced low at the channel (n) match when counting up. See the following figure.

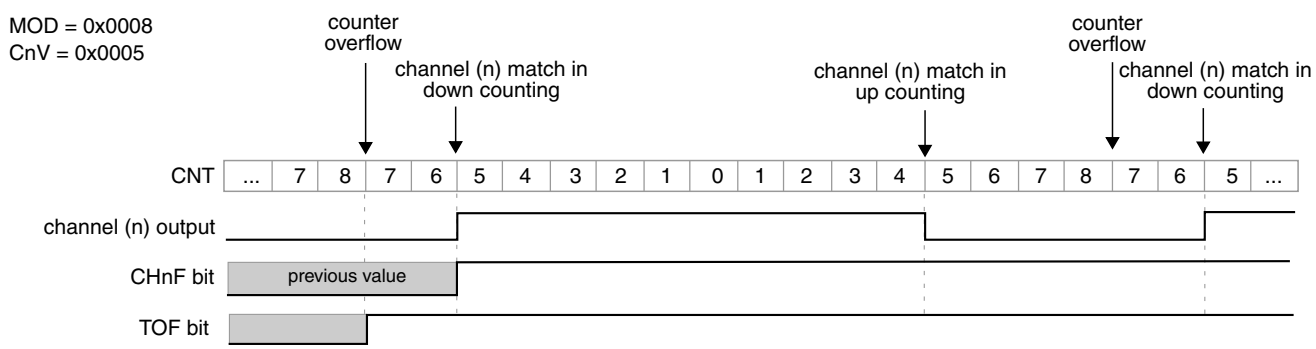


Figure 36-23. CPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the channel (n) match (FTM counter = CnV) when counting down, and it is forced high at the channel (n) match when counting up. See the following figure.

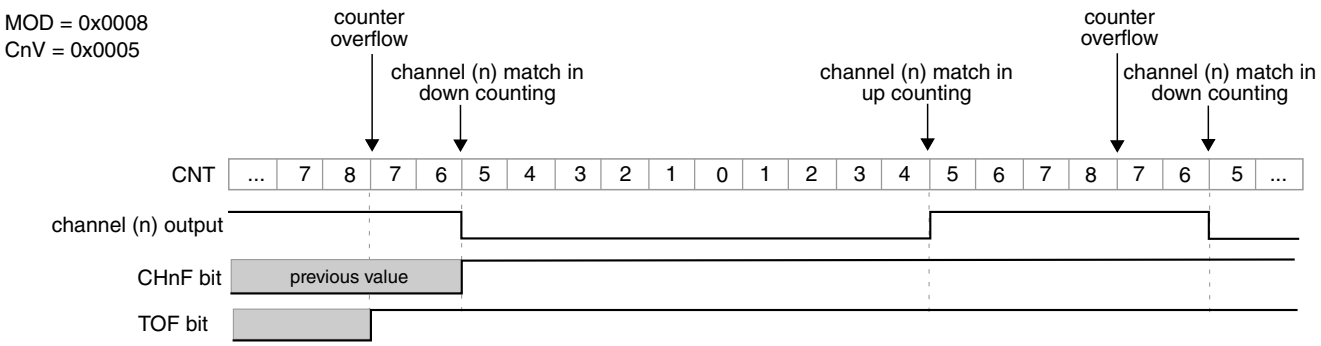


Figure 36-24. CPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000) or CnV is a negative value, that is (CnV[15] = 1), then the channel (n) output is a 0% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match.

If CnV is a positive value, that is (CnV[15] = 0), (CnV ≥ MOD), and (MOD ≠ 0x0000), then the channel (n) output is a 100% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match. This implies that the usable range of periods set by MOD is 0x0001 through 0x7FFE, 0x7FFF if you do not need to generate a 100% duty cycle CPWM signal. This is not a significant limitation because the resulting period is much longer than required for normal applications.

The CPWM mode must not be used when the FTM counter is a free running counter.

36.4.8 Combine mode

The Combine mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMBINE = 1, and
- CPWMS = 0

In Combine mode, an even channel (n) and adjacent odd channel (n+1) are combined to generate a PWM signal in the channel (n) output.

In the Combine mode, the PWM period is determined by (MOD − CNTIN + 0x0001) and the PWM pulse width (duty cycle) is determined by (|C(n+1)V − C(n)V|).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = C(n)V). The CH(n+1)F bit is set and the channel (n +1) interrupt is generated, if CH(n+1)IE = 1, at the channel (n+1) match (FTM counter = C(n+1)V).

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n +1)V). It is forced high at the channel (n) match (FTM counter = C(n)V). See the following figure.

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced high at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n +1)V). It is forced low at the channel (n) match (FTM counter = C(n)V). See the following figure.

In Combine mode, the ELS(n+1)B and ELS(n+1)A bits are not used in the generation of the channels (n) and (n+1) output. However, if (ELSnB:ELSnA = 0:0) then the channel (n) output is not controlled by FTM, and if (ELS(n+1)B:ELS(n+1)A = 0:0) then the channel (n+1) output is not controlled by FTM.

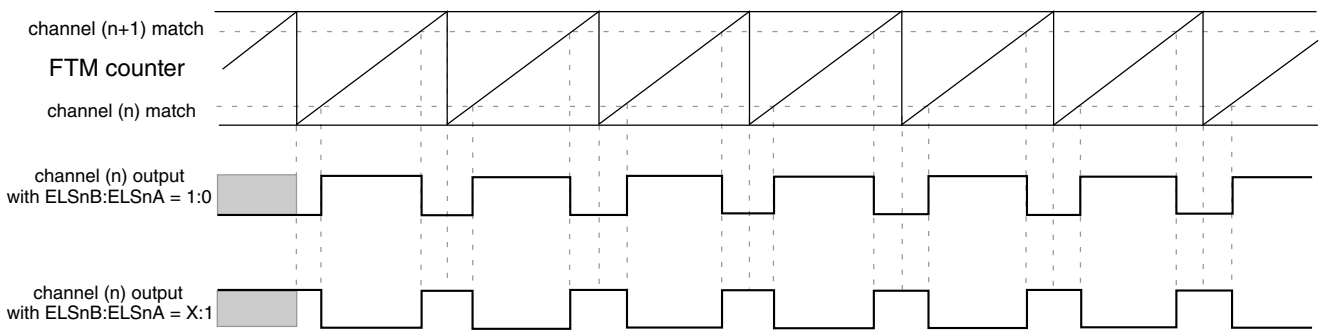


Figure 36-25. Combine mode

The following figures illustrate the PWM signals generation using Combine mode.

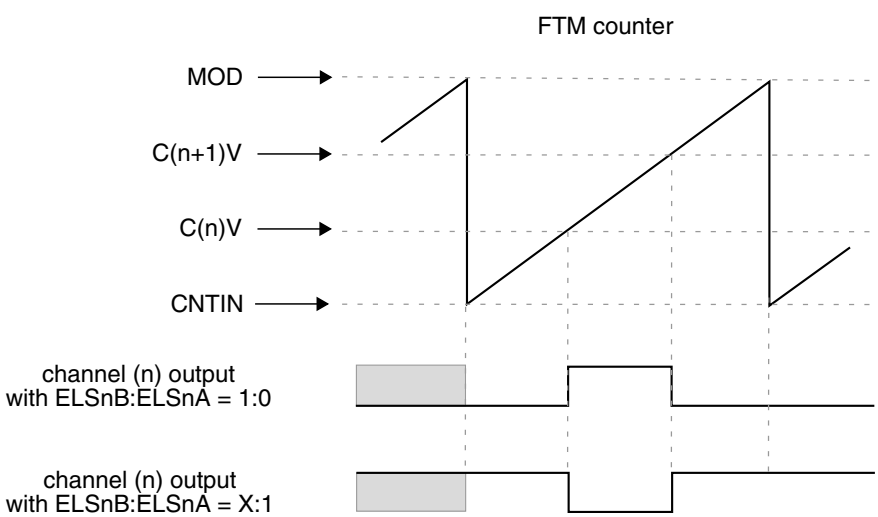


Figure 36-26. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V < C(n+1)V)

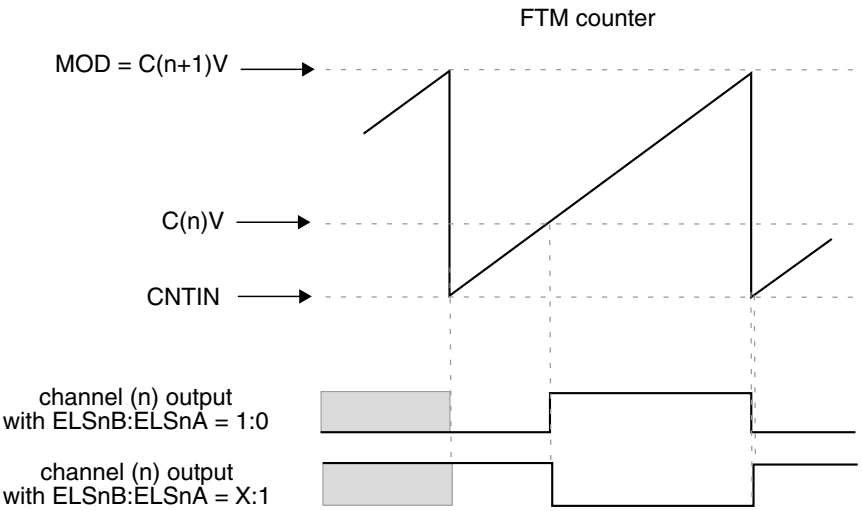


Figure 36-27. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n+1)V = MOD)

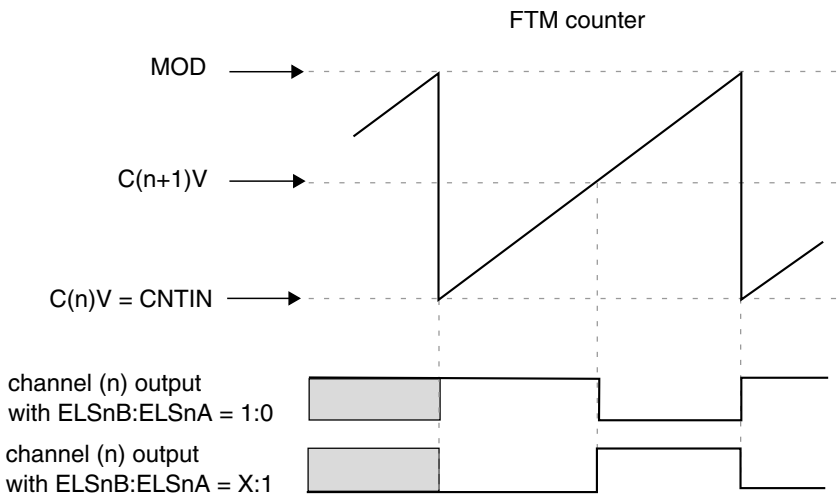


Figure 36-28. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD)

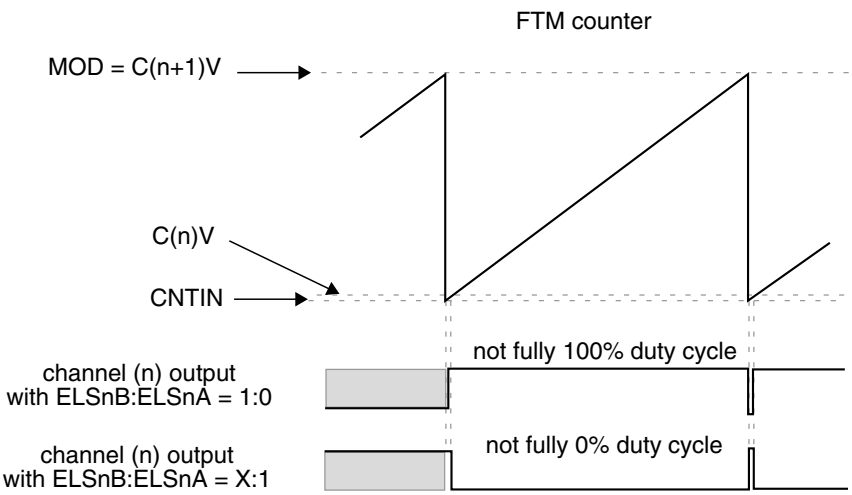


Figure 36-29. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n)V is Almost Equal to CNTIN) and (C(n+1)V = MOD)

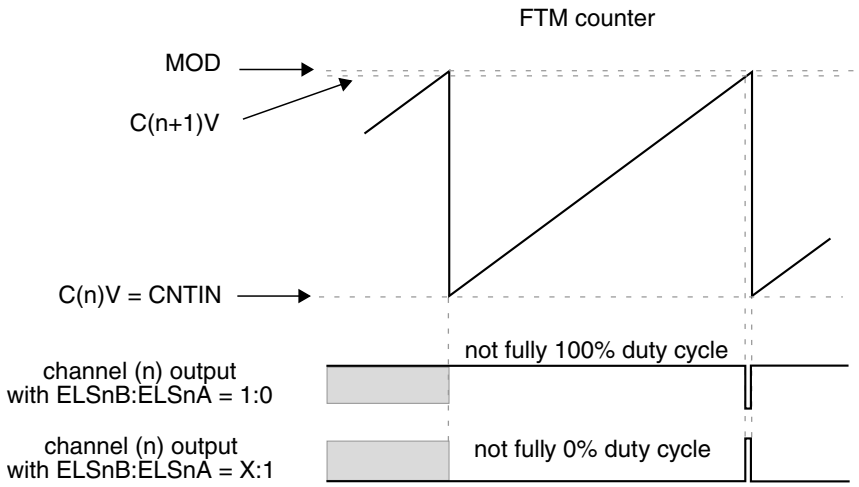


Figure 36-30. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD) and (C(n+1)V is Almost Equal to MOD)

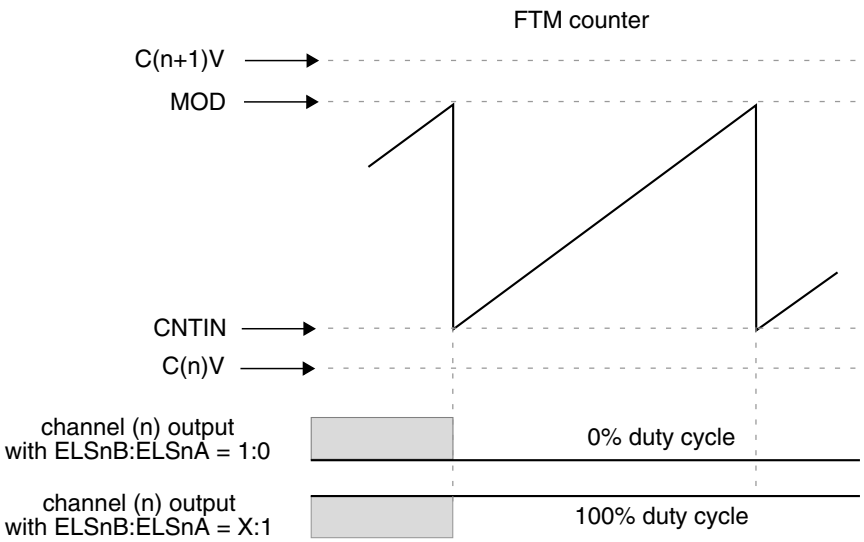


Figure 36-31. Channel (n) output if C(n)V and C(n+1)V are not between CNTIN and MOD

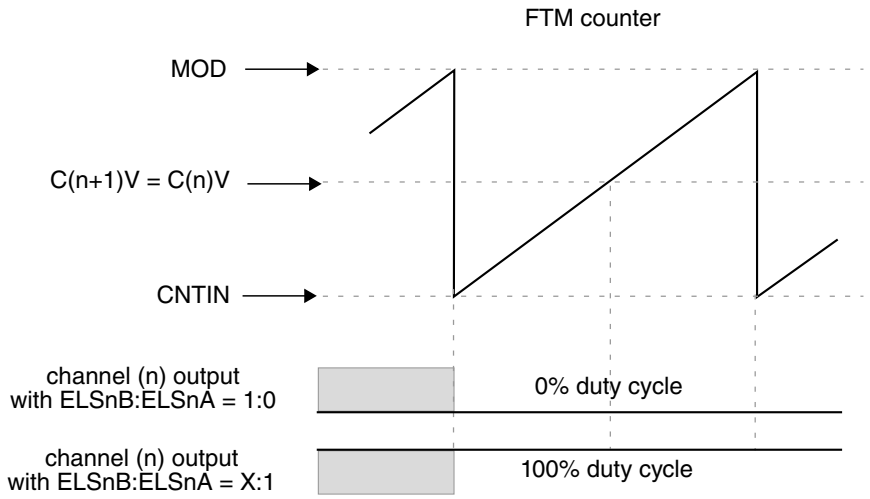


Figure 36-32. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V = C(n+1)V)

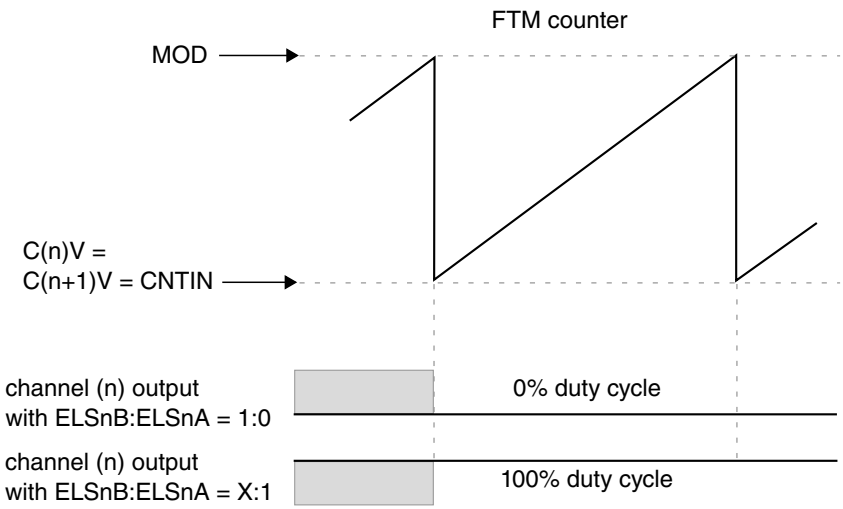


Figure 36-33. Channel (n) output if (C(n)V = C(n+1)V = CNTIN)

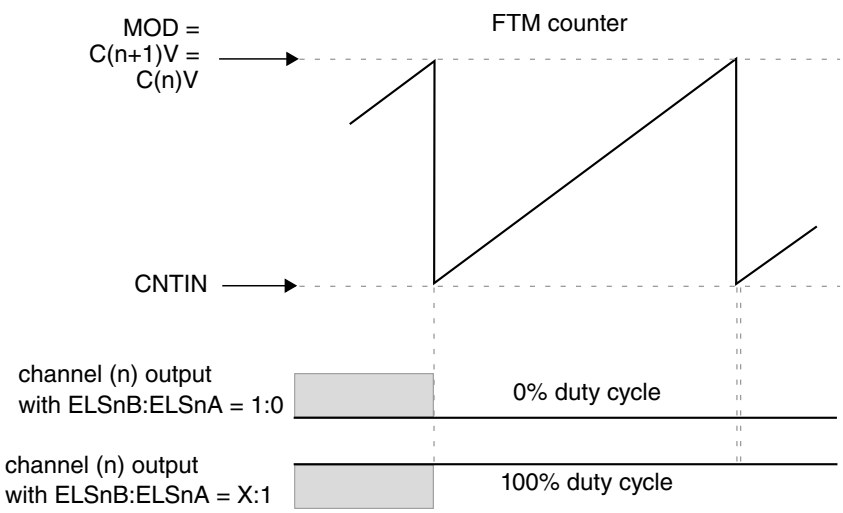


Figure 36-34. Channel (n) output if (C(n)V = C(n+1)V = MOD)

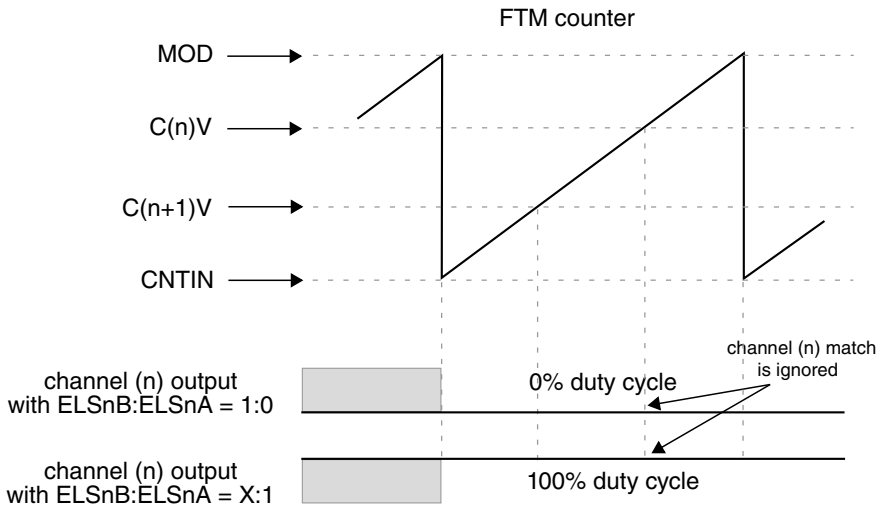


Figure 36-35. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V > C(n+1)V)

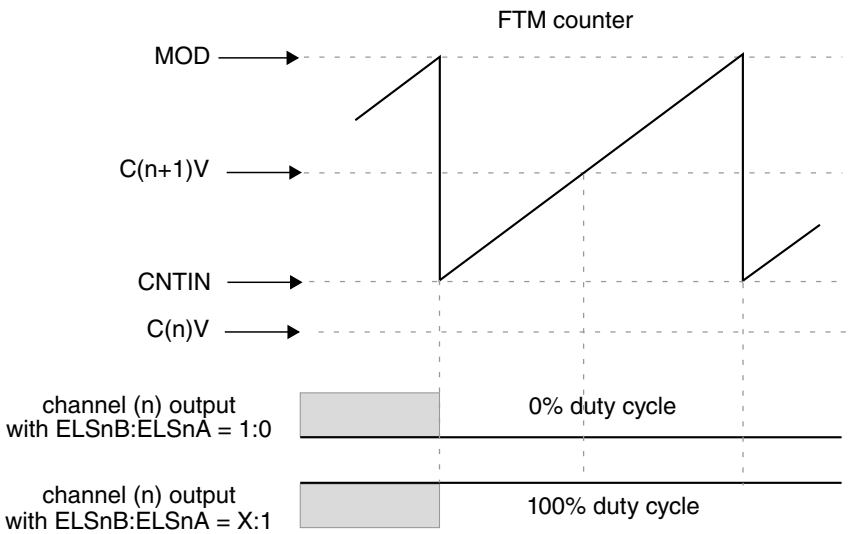


Figure 36-36. Channel (n) output if (C(n)V < CNTIN) and (CNTIN < C(n+1)V < MOD)

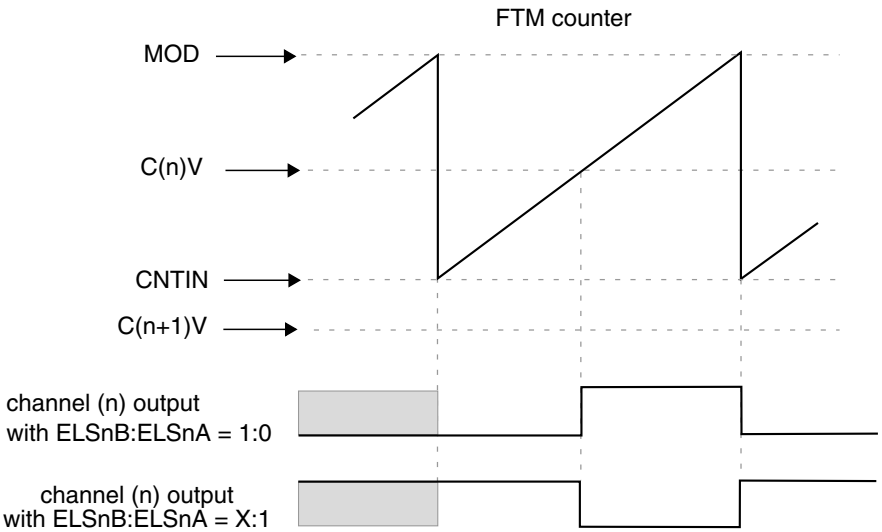


Figure 36-37. Channel (n) output if (C(n+1)V < CNTIN) and (CNTIN < C(n)V < MOD)

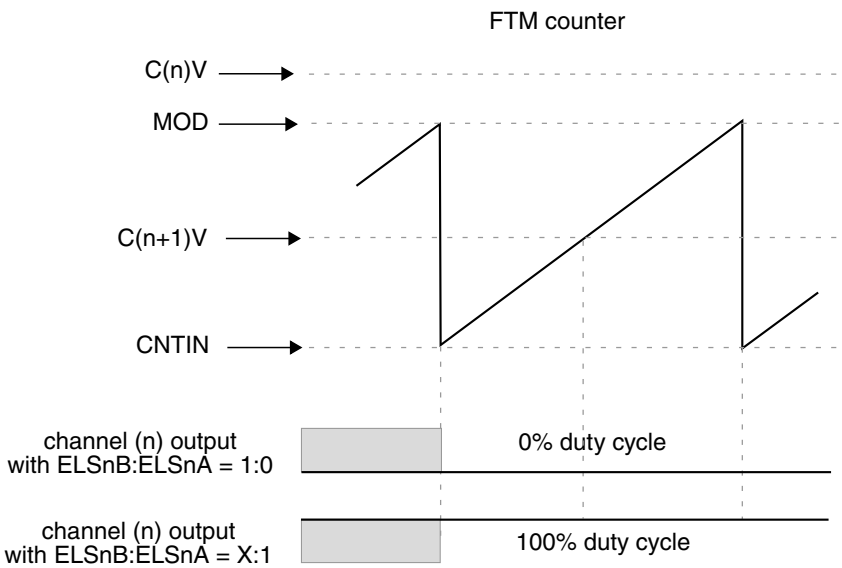


Figure 36-38. Channel (n) output if (C(n)V > MOD) and (CNTIN < C(n+1)V < MOD)

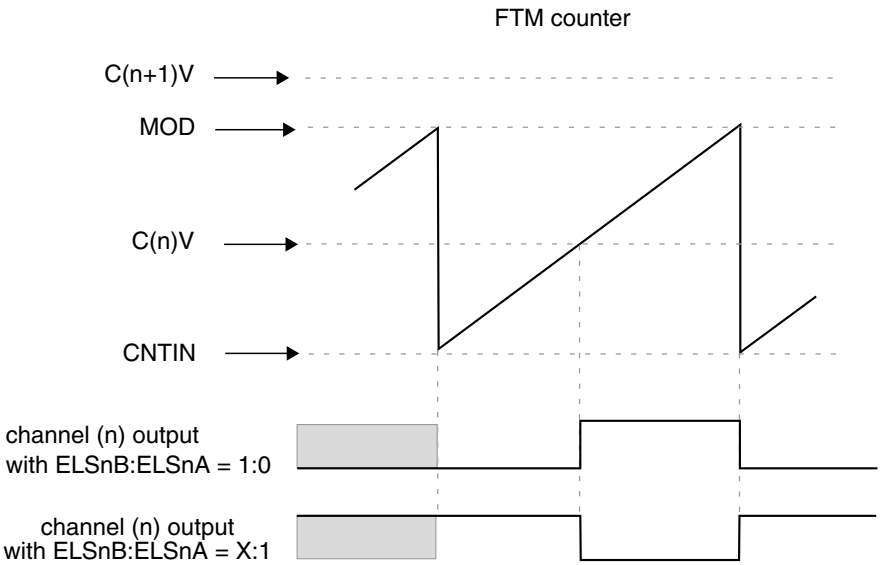


Figure 36-39. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V < MOD)

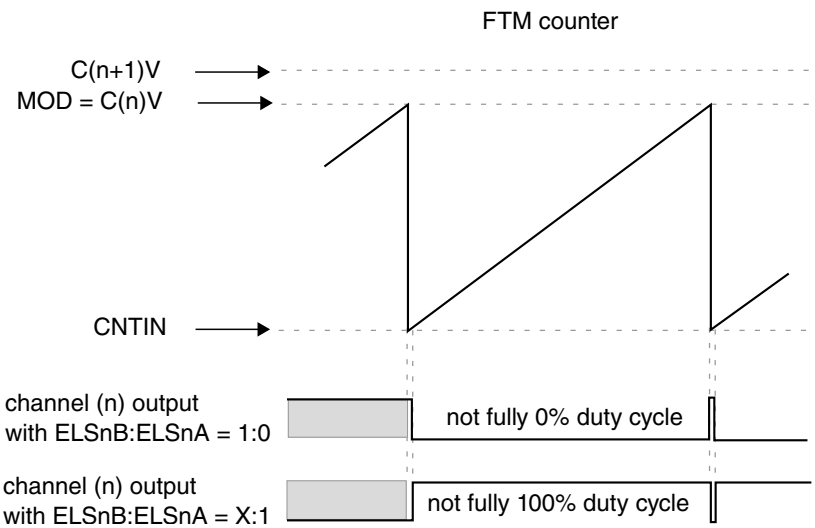


Figure 36-40. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V = MOD)

36.4.8.1 Asymmetrical PWM

In Combine mode, the control of the PWM signal first edge, when the channel (n) match occurs, that is, FTM counter = C(n)V, is independent of the control of the PWM signal second edge, when the channel (n+1) match occurs, that is, FTM counter = C(n+1)V. So, Combine mode allows the generation of asymmetrical PWM signals.

36.4.9 Complementary mode

The Complementary mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 1

In Complementary mode, the channel (n+1) output is the inverse of the channel (n) output.

So, the channel (n+1) output is the same as the channel (n) output when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 0

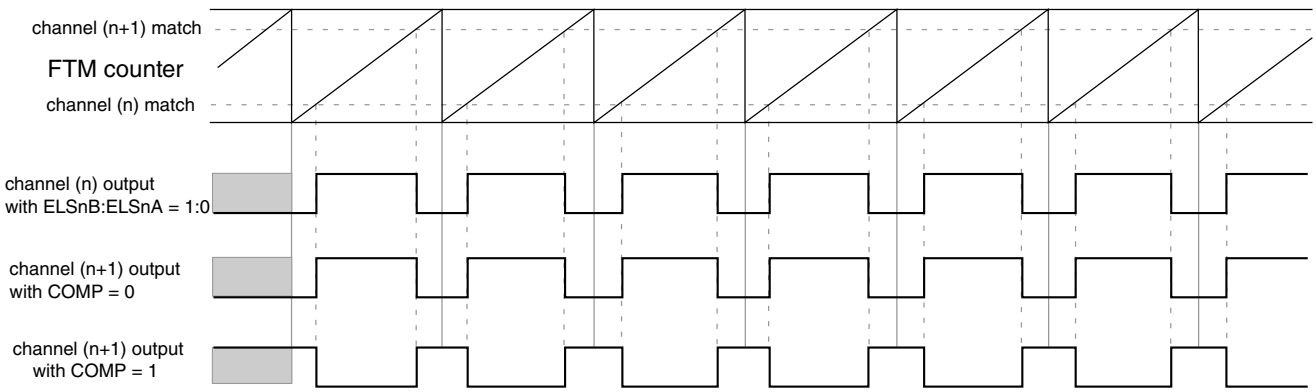


Figure 36-41. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = 1:0)

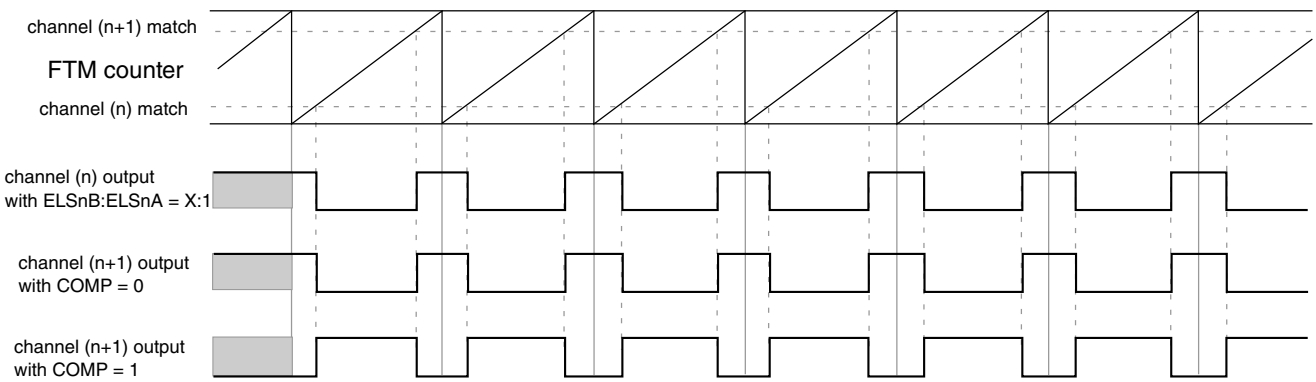


Figure 36-42. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = X:1)

NOTE

The complementary mode is not available in Output Compare mode.

36.4.10 Registers updated from write buffers

36.4.10.1 CNTIN register update

The following table describes when CNTIN register is updated:

Table 36-5. CNTIN register update

When	Then CNTIN register is updated
CLKS[1:0] = 0:0	When CNTIN register is written, independent of FTMEN bit.
• FTMEN = 0, or • CNTINC = 0	At the next system clock after CNTIN was written.
• FTMEN = 1, • SYNCMODE = 1, and • CNTINC = 1	By the CNTIN register synchronization.

36.4.10.2 MOD register update

The following table describes when MOD register is updated:

Table 36-6. MOD register update

When	Then MOD register is updated
CLKS[1:0] = 0:0	When MOD register is written, independent of FTMEN bit.
• CLKS[1:0] ≠ 0:0, and • FTMEN = 0	According to the CPWMS bit, that is: • If the selected mode is not CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000. • If the selected mode is CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to (MOD – 0x0001).
• CLKS[1:0] ≠ 0:0, and • FTMEN = 1	By the MOD register synchronization.

36.4.10.3 CnV register update

The following table describes when CnV register is updated:

Table 36-7. CnV register update

When	Then CnV register is updated
CLKS[1:0] = 0:0	When CnV register is written, independent of FTMEN bit.
• CLKS[1:0] ≠ 0:0, and • FTMEN = 0	According to the selected mode, that is:

Table continues on the next page...

Table 36-7. CnV register update (continued)

When	Then CnV register is updated
	• If the selected mode is Output Compare, then CnV register is updated on the next FTM counter change, end of the prescaler counting, after CnV register was written. • If the selected mode is EPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000. • If the selected mode is CPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to (MOD – 0x0001).
• CLKS[1:0] ≠ 0:0, and • FTMEN = 1	According to the selected mode, that is: • If the selected mode is output compare then CnV register is updated according to the SYNCEN bit. If (SYNCEN = 0) then CnV register is updated after CnV register was written at the next change of the FTM counter, the end of the prescaler counting. If (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization. • If the selected mode is not output compare and (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization.

36.4.11 PWM synchronization

The PWM synchronization provides an opportunity to update the MOD, CNTIN, CnV, OUTMASK, INVCTRL and SWOCTRL registers with their buffered value and force the FTM counter to the CNTIN register value.

Note

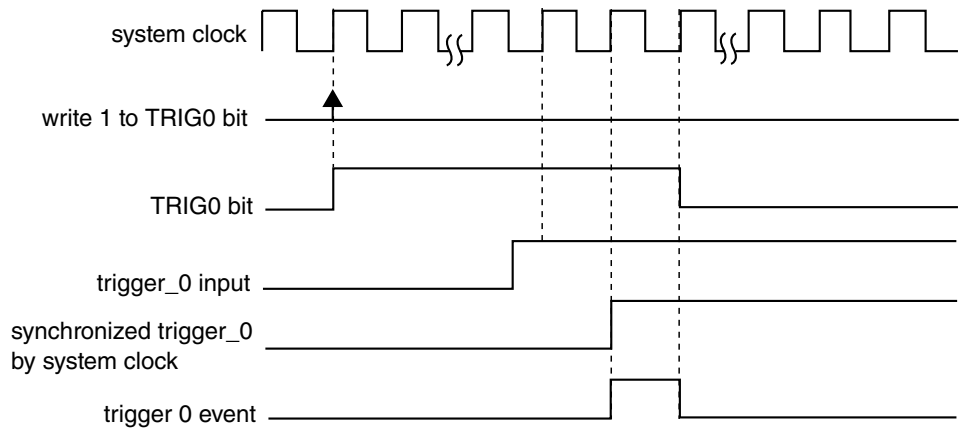
The legacy PWM synchronization (SYNCMODE = 0) is a subset of the enhanced PWM synchronization (SYNCMODE = 1). Thus, only the enhanced PWM synchronization must be used.

36.4.11.1 Hardware trigger

Three hardware trigger signal inputs of the FTM module are enabled when TRIGn = 1, where n = 0, 1 or 2 corresponding to each one of the input signals, respectively. The hardware trigger input n is synchronized by the system clock. The PWM synchronization with hardware trigger is initiated when a rising edge is detected at the enabled hardware trigger inputs.

If (HWTRIGMODE = 0) then the TRIGn bit is cleared when 0 is written to it or when the trigger n event is detected.

In this case, if two or more hardware triggers are enabled (for example, TRIG0 and TRIG1 = 1) and only trigger 1 event occurs, then only TRIG1 bit is cleared. If a trigger n event occurs together with a write setting TRIGn bit, then the synchronization is initiated, but TRIGn bit remains set due to the write operation.



Note
All hardware trigger inputs have the same behavior.

Figure 36-43. Hardware trigger event with HWTRIGMODE = 0

If HWTRIGMODE = 1, then the TRIGn bit is only cleared when 0 is written to it.

NOTE

The HWTRIGMODE bit must be 1 only with enhanced PWM synchronization (SYNCMODE = 1).

36.4.11.2 Software trigger

A software trigger event occurs when 1 is written to the SYNC[SWSYNC] bit. The SWSYNC bit is cleared when 0 is written to it or when the PWM synchronization, initiated by the software event, is completed.

If another software trigger event occurs (by writing another 1 to the SWSYNC bit) at the same time the PWM synchronization initiated by the previous software trigger event is ending, a new PWM synchronization is started and the SWSYNC bit remains equal to 1.

If SYNCMODE = 0 then the SWSYNC bit is also cleared by FTM according to PWMSYNC and REINIT bits. In this case if (PWMSYNC = 1) or (PWMSYNC = 0 and REINIT = 0) then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see Boundary cycle and loading points and the following figure. If (PWMSYNC = 0) and (REINIT = 1) then SWSYNC bit is cleared when the software trigger event occurs.

If SYNCMODE = 1 then the SWSYNC bit is also cleared by FTM according to the SWRSTCNT bit. If SWRSTCNT = 0 then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see the following figure. If SWRSTCNT = 1 then SWSYNC bit is cleared when the software trigger event occurs.

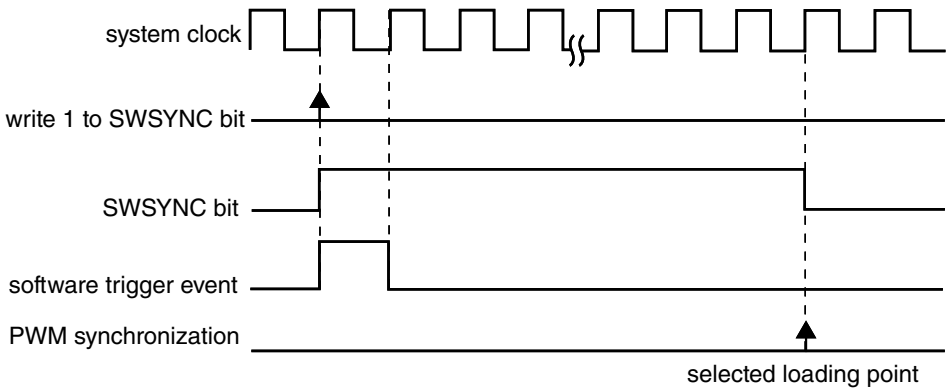


Figure 36-44. Software trigger event

36.4.11.3 Boundary cycle and loading points

The boundary cycle definition is important for the loading points for the registers MOD, CNTIN, and C(n)V.

In Up counting mode, the boundary cycle is defined as when the counter wraps to its initial value (CNTIN). If in Up-down counting mode, then the boundary cycle is defined as when the counter turns from down to up counting and when from up to down counting.

The following figure shows the boundary cycles and the loading points for the registers. In the Up Counting mode, the loading points are enabled if one of CNTMIN or CTMAX bits are 1. In the Up-Down Counting mode, the loading points are selected by CNTMIN and CNTMAX bits, as indicated in the figure. These loading points are safe places for register updates thus allowing a smooth transitions in PWM waveform generation.

For both counting modes, if neither CNTMIN nor CNTMAX are 1, then the boundary cycles are not used as loading points for registers updates. See the register synchronization descriptions in the following sections for details.

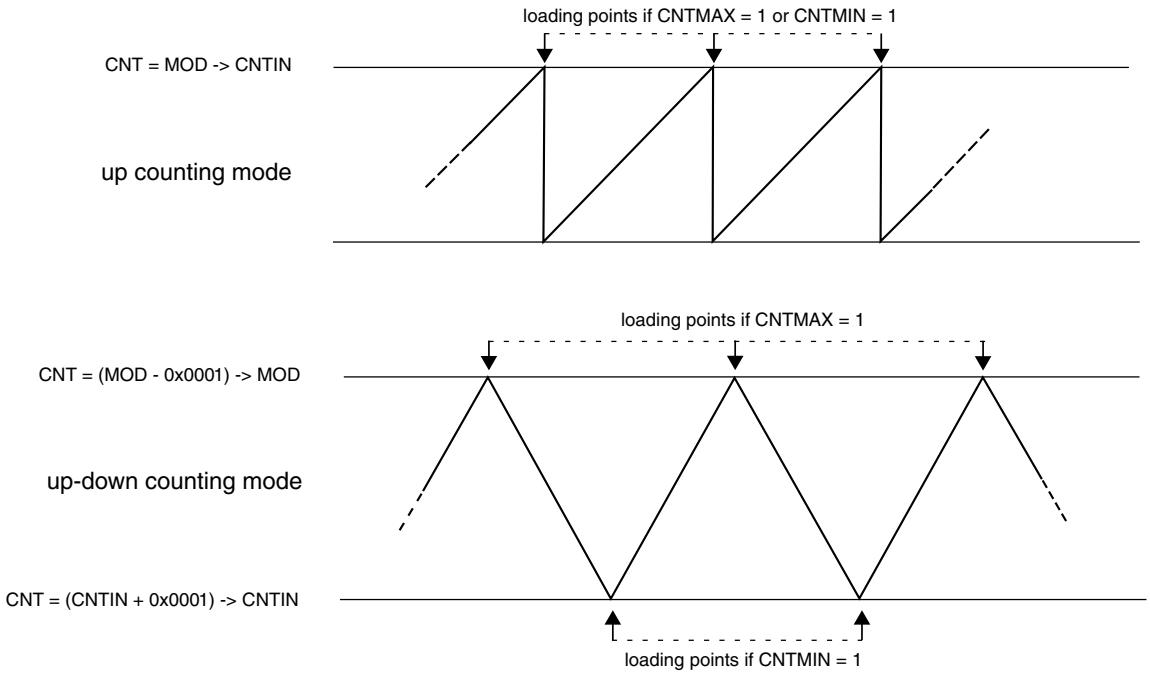


Figure 36-45. Boundary cycles and loading points

36.4.11.4 MOD register synchronization

The MOD register synchronization updates the MOD register with its buffer value. This synchronization is enabled if (FTMEN = 1).

The MOD register synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, it is expected that the MOD register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the MOD register synchronization depends on SWWRBUF, SWRSTCNT, HWWRBUF, and HWRSTCNT bits according to this flowchart:

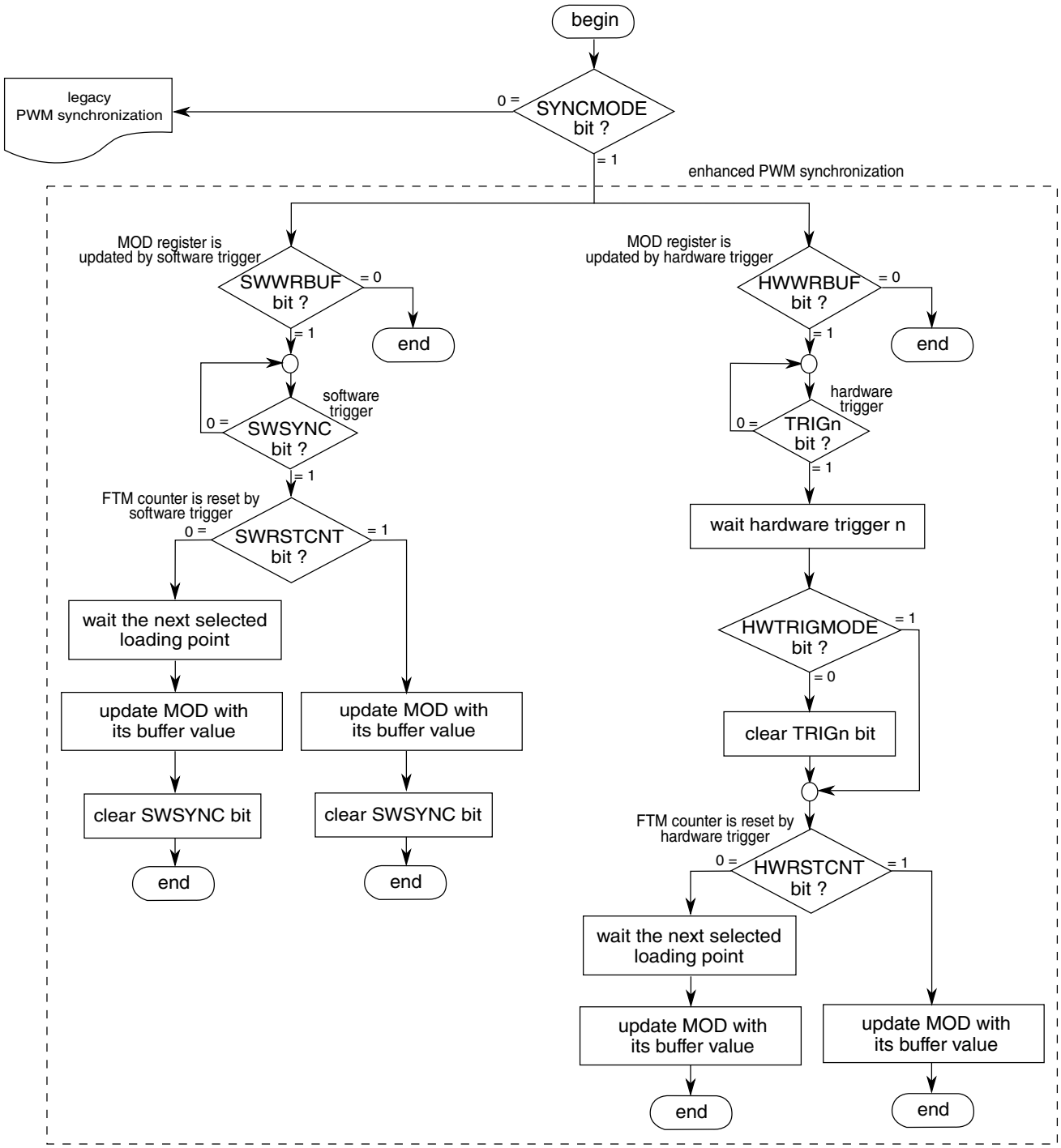


Figure 36-46. MOD register synchronization flowchart

In the case of legacy PWM synchronization, the MOD register synchronization depends on PWMSYNC and REINIT bits according to the following description.

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 0), then this synchronization is made on the next selected loading point after an enabled trigger event takes place. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected

loading point. If the trigger event was a hardware trigger, then the trigger enable bit (TRIGn) is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

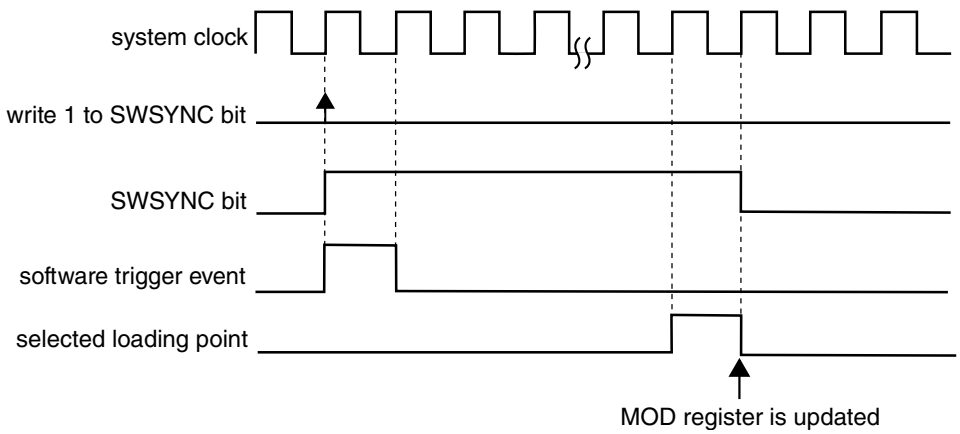


Figure 36-47. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 0), and software trigger was used

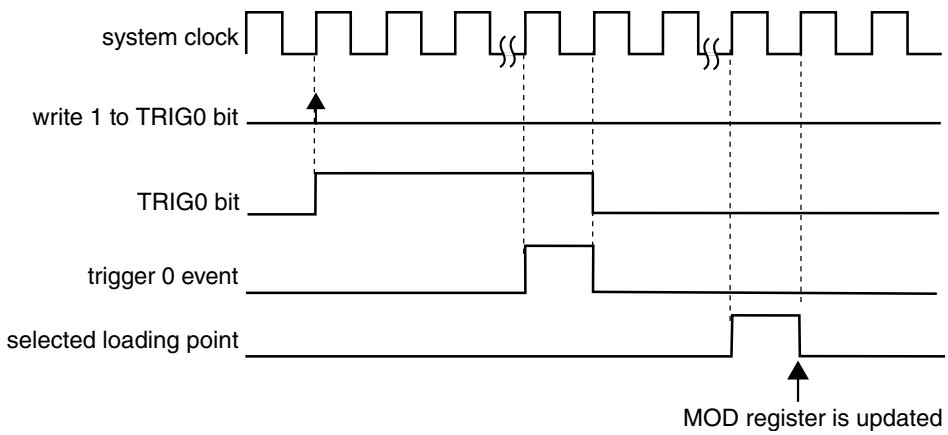


Figure 36-48. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 0), and a hardware trigger was used

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 1), then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

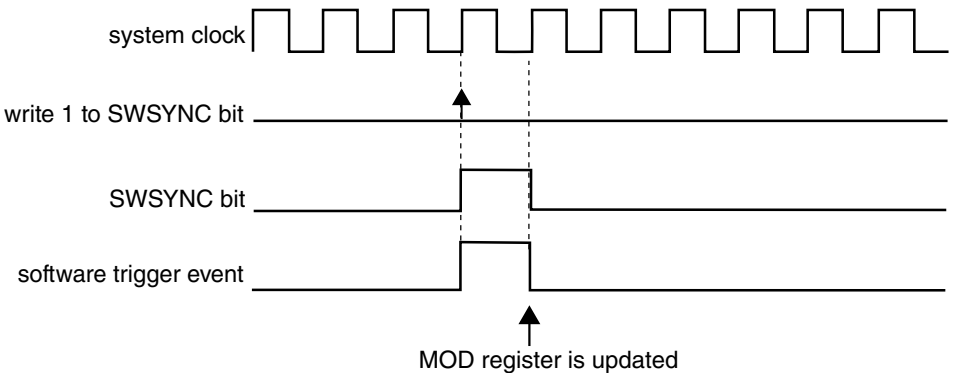


Figure 36-49. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 1), and software trigger was used

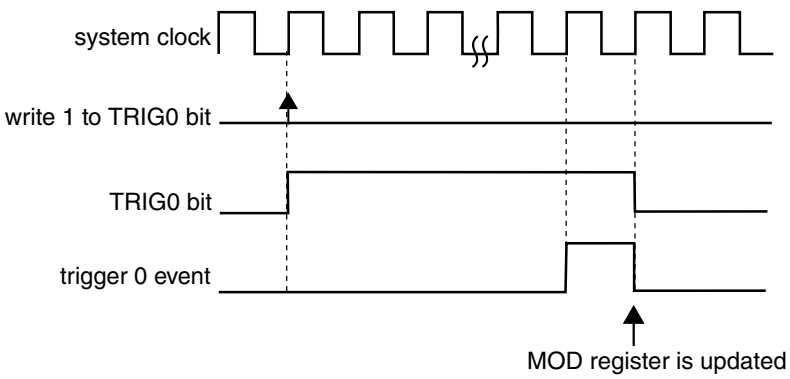


Figure 36-50. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 1), and a hardware trigger was used

If (SYNCMODE = 0) and (PWMSYNC = 1), then this synchronization is made on the next selected loading point after the software trigger event takes place. The SWSYNC bit is cleared on the next selected loading point:

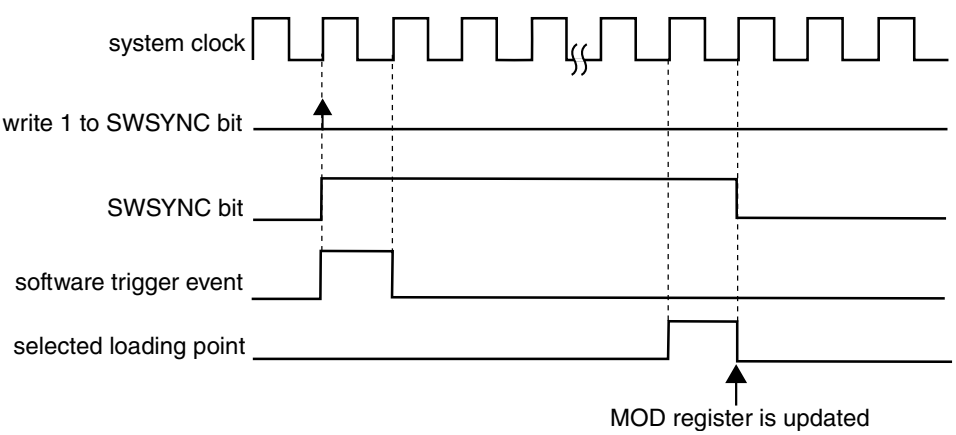


Figure 36-51. MOD synchronization with (SYNCMODE = 0) and (PWMSYNC = 1)

36.4.11.5 CNTIN register synchronization

The CNTIN register synchronization updates the CNTIN register with its buffer value.

This synchronization is enabled if (FTMEN = 1), (SYNCMODE = 1), and (CNTINC = 1). The CNTIN register synchronization can be done only by the enhanced PWM synchronization (SYNCMODE = 1). The synchronization mechanism is the same as the MOD register synchronization done by the enhanced PWM synchronization; see MOD register synchronization.

36.4.11.6 C(n)V and C(n+1)V register synchronization

The C(n)V and C(n+1)V registers synchronization updates the C(n)V and C(n+1)V registers with their buffer values.

This synchronization is enabled if (FTMEN = 1) and (SYNCEN = 1). The synchronization mechanism is the same as the MOD register synchronization. However, it is expected that the C(n)V and C(n+1)V registers be synchronized only by the enhanced PWM synchronization (SYNCMODE = 1).

36.4.11.7 OUTMASK register synchronization

The OUTMASK register synchronization updates the OUTMASK register with its buffer value.

The OUTMASK register can be updated at each rising edge of system clock (SYNCHOM = 0), by the enhanced PWM synchronization (SYNCHOM = 1 and SYNCMODE = 1) or by the legacy PWM synchronization (SYNCHOM = 1 and SYNCMODE = 0). However, it is expected that the OUTMASK register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the OUTMASK register synchronization depends on SWOM and HWOM bits. See the following flowchart:

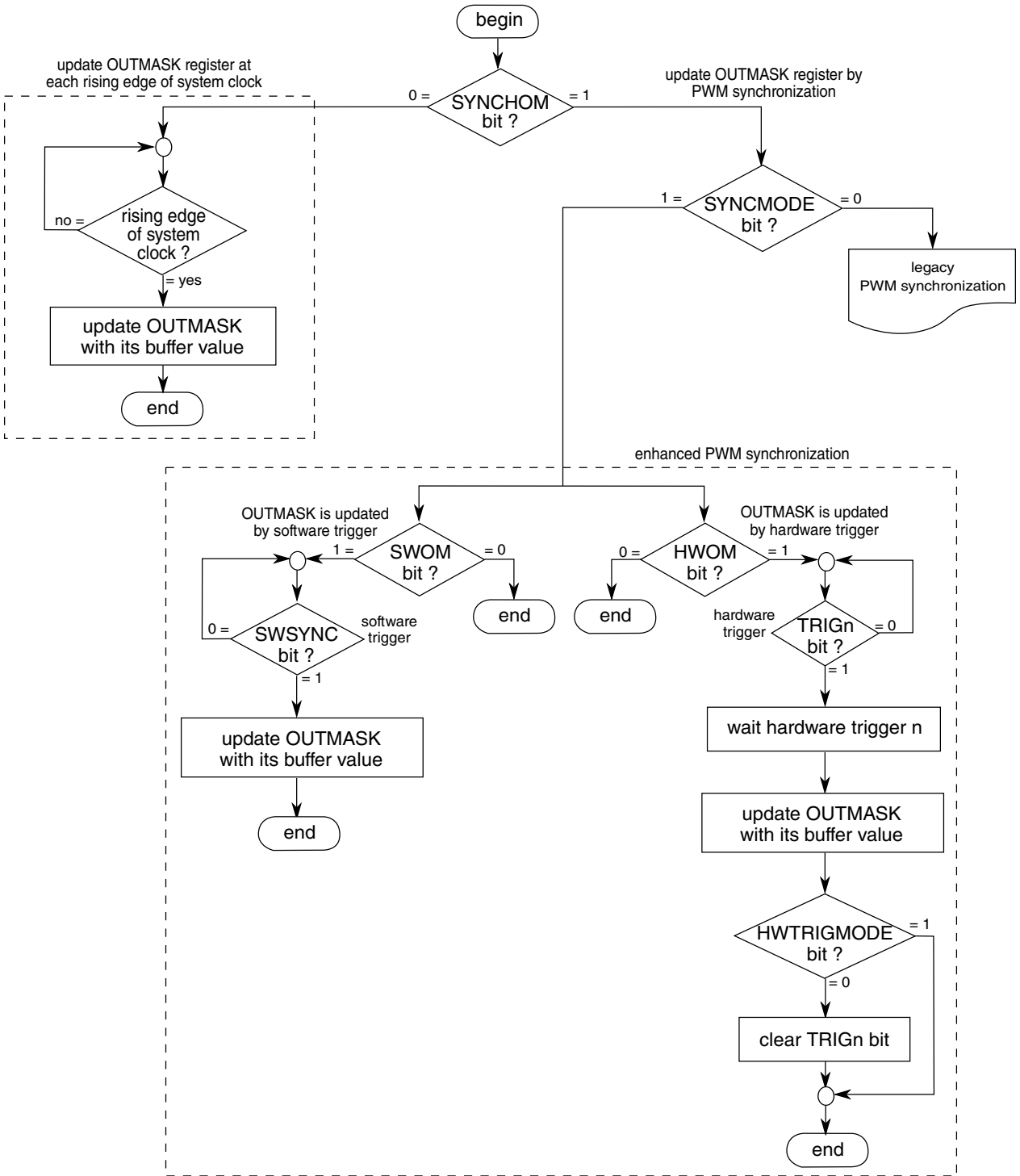


Figure 36-52. OUTMASK register synchronization flowchart

In the case of legacy PWM synchronization, the OUTMASK register synchronization depends on PWMSYNC bit according to the following description.

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 0), then this synchronization is done on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected loading point. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

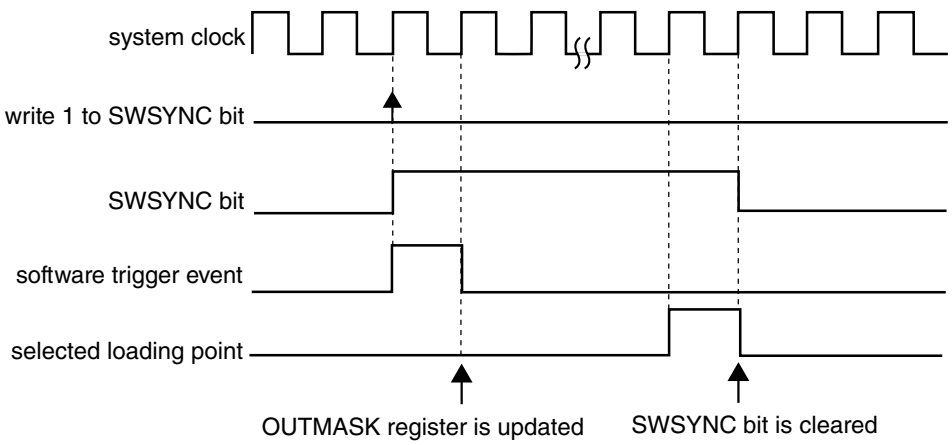


Figure 36-53. OUTMASK synchronization with (SYNCMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0) and software trigger was used

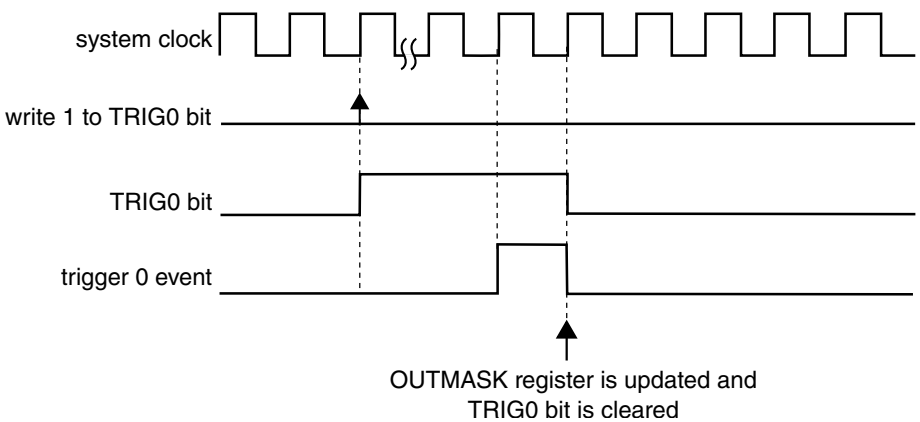


Figure 36-54. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0), and a hardware trigger was used

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 1), then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger. An example with a hardware trigger follows.

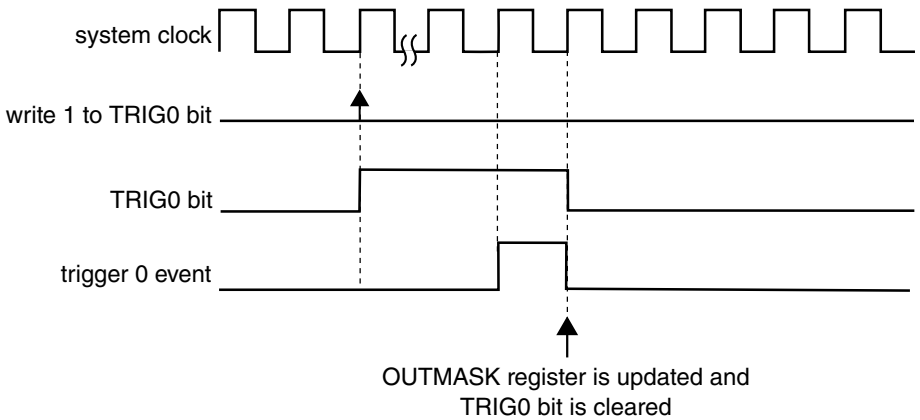


Figure 36-55. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 1), and a hardware trigger was used

36.4.11.8 INVCTRL register synchronization

The INVCTRL register synchronization updates the INVCTRL register with its buffer value.

The INVCTRL register can be updated at each rising edge of system clock (INVC = 0) or by the enhanced PWM synchronization (INVC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the INVCTRL register synchronization depends on SWINVC and HWINVC bits.

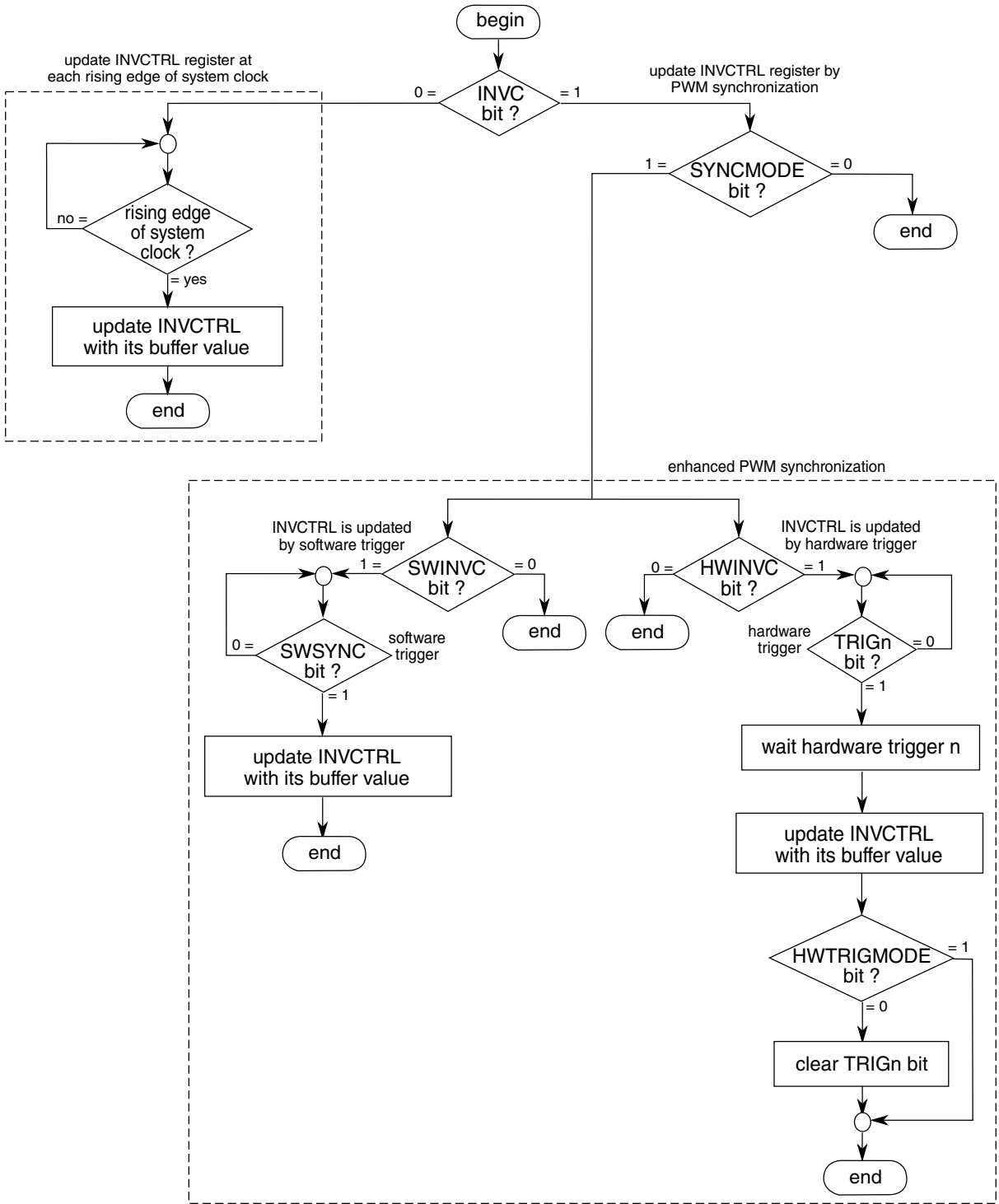


Figure 36-56. INVCTRL register synchronization flowchart

36.4.11.9 SWOCTRL register synchronization

The SWOCTRL register synchronization updates the SWOCTRL register with its buffer value.

The SWOCTRL register can be updated at each rising edge of system clock (SWOC = 0) or by the enhanced PWM synchronization (SWOC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the SWOCTRL register synchronization depends on SWSOC and HWSOC bits.

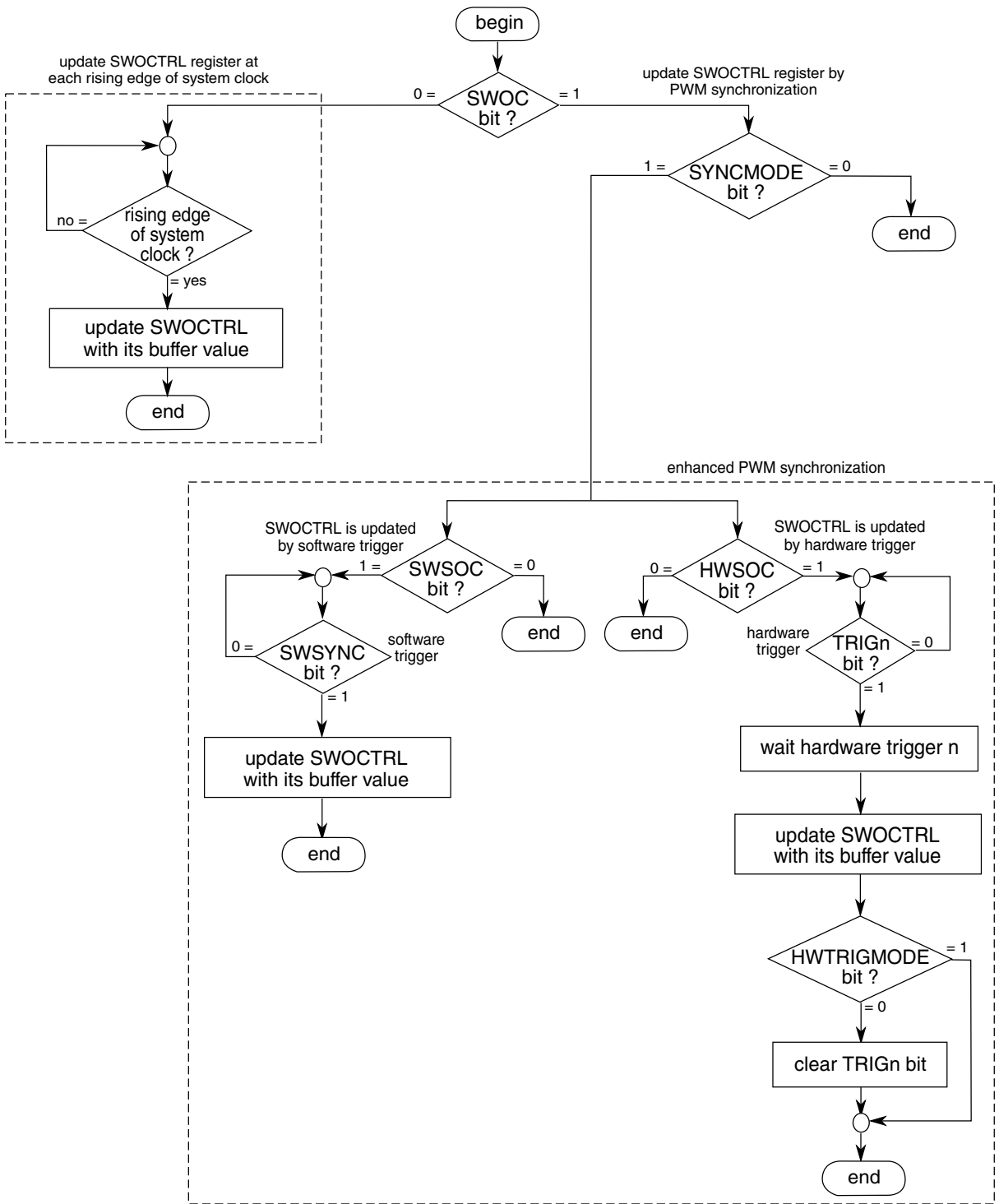


Figure 36-57. SWOCTRL register synchronization flowchart

36.4.11.10 FTM counter synchronization

The FTM counter synchronization is a mechanism that allows the FTM to restart the PWM generation at a certain point in the PWM period. The channels outputs are forced to their initial value, except for channels in Output Compare mode, and the FTM counter is forced to its initial counting value defined by CNTIN register.

The following figure shows the FTM counter synchronization. Note that after the synchronization event occurs, the channel (n) is set to its initial value and the channel (n +1) is not set to its initial value due to a specific timing of this figure in which the deadtime insertion prevents this channel output from transitioning to 1. If no deadtime insertion is selected, then the channel (n+1) transitions to logical value 1 immediately after the synchronization event occurs.

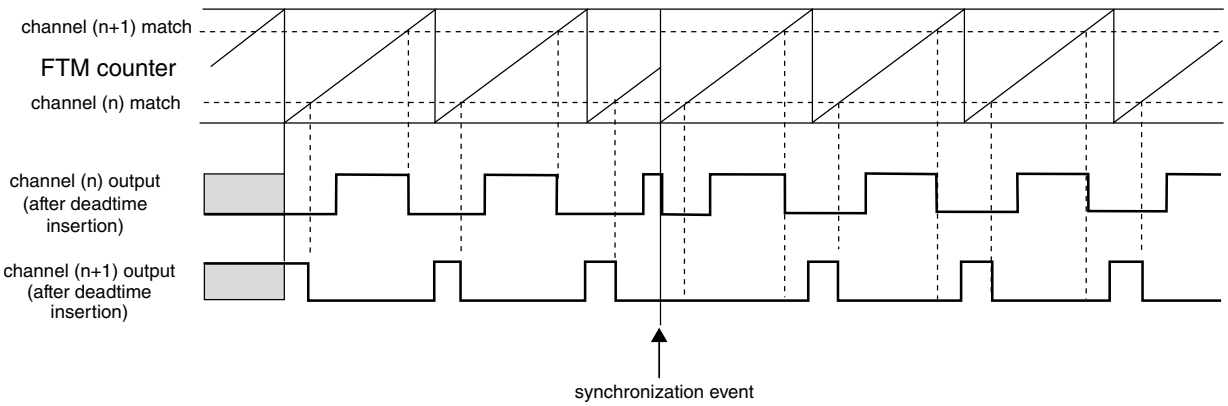


Figure 36-58. FTM counter synchronization

The FTM counter synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, the FTM counter must be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the FTM counter synchronization depends on SWRSTCNT and HWRSTCNT bits according to the following flowchart.

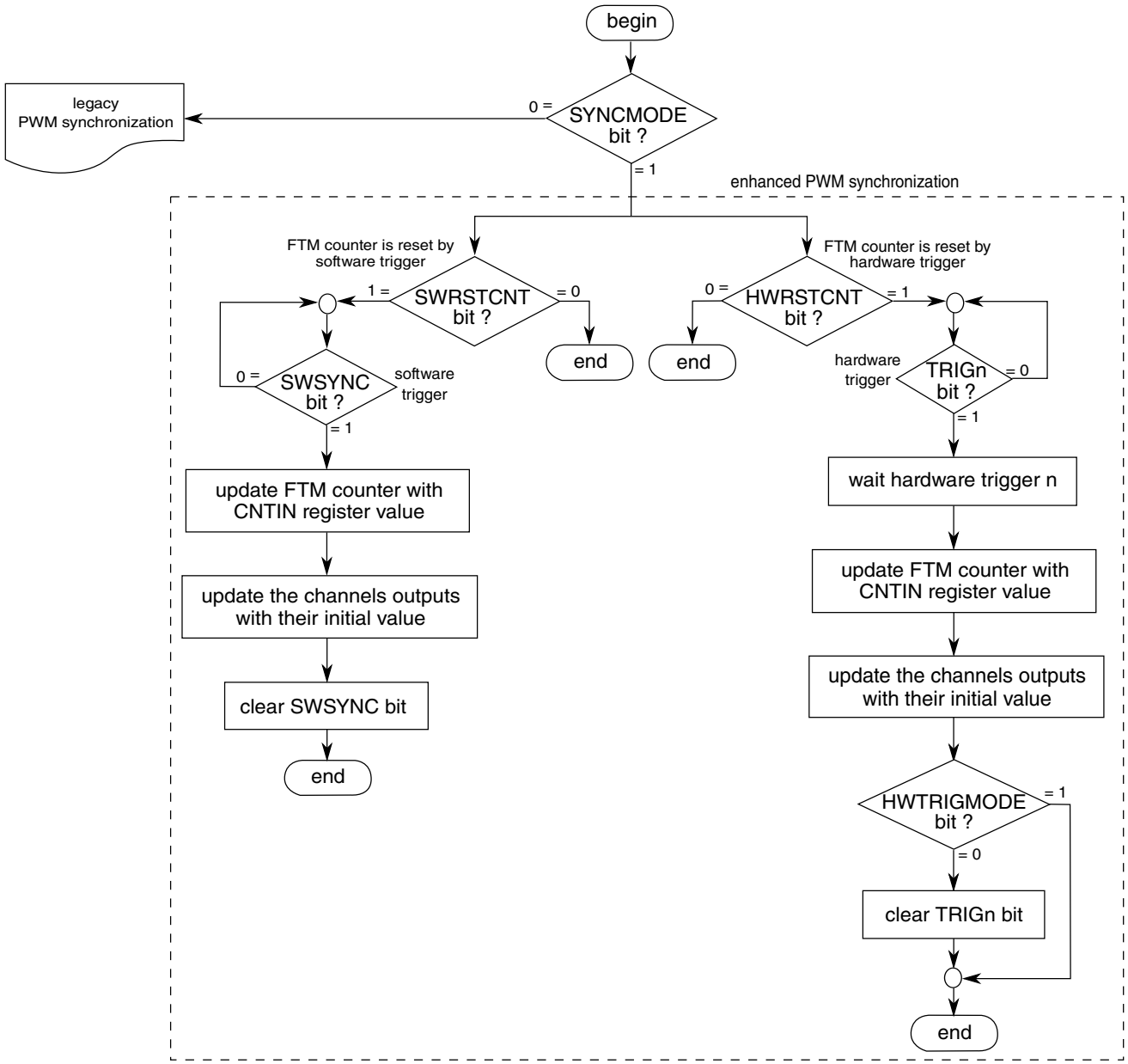


Figure 36-59. FTM counter synchronization flowchart

In the case of legacy PWM synchronization, the FTM counter synchronization depends on REINIT and PWMSYNC bits according to the following description.

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 0) then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

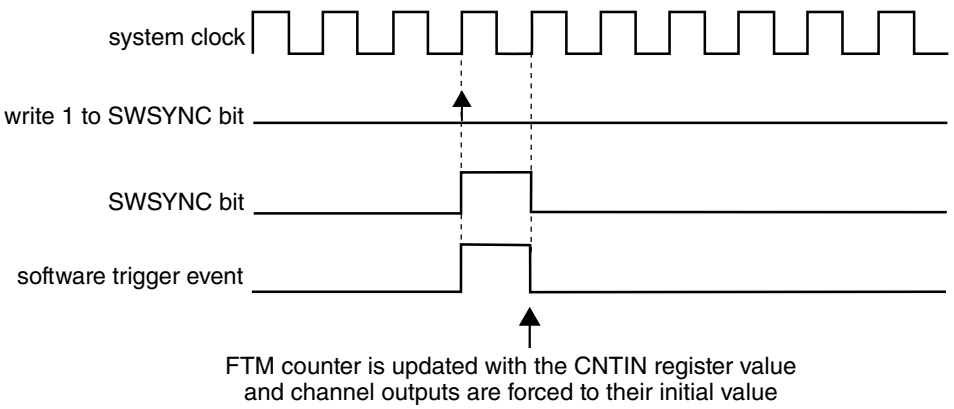


Figure 36-60. FTM counter synchronization with (SYNCMODE = 0), (REINIT = 1), (PWMSYNC = 0), and software trigger was used

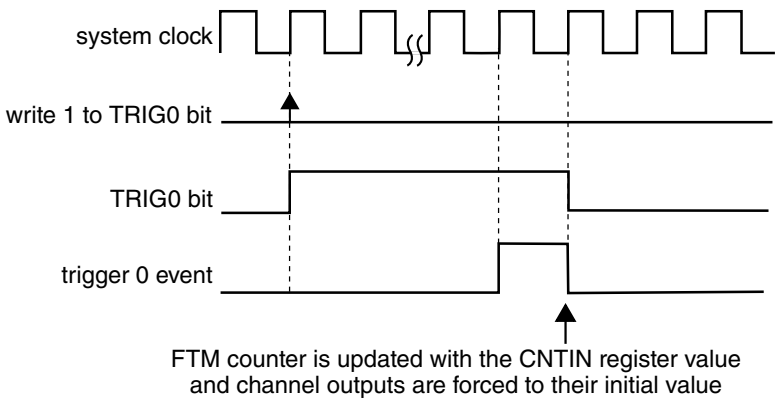


Figure 36-61. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 0), and a hardware trigger was used

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 1) then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger.

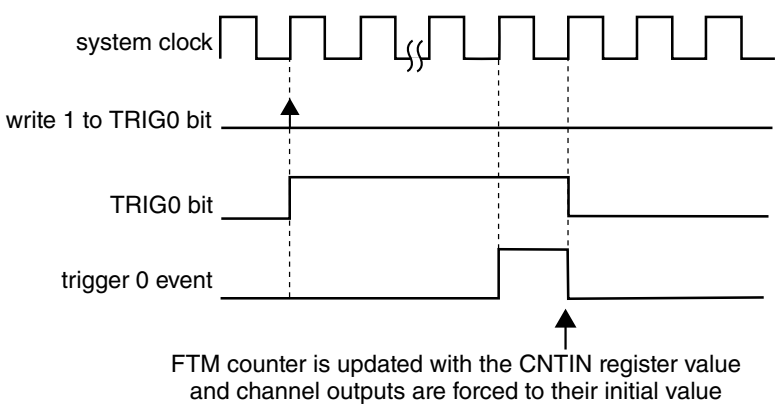


Figure 36-62. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 1), and a hardware trigger was used

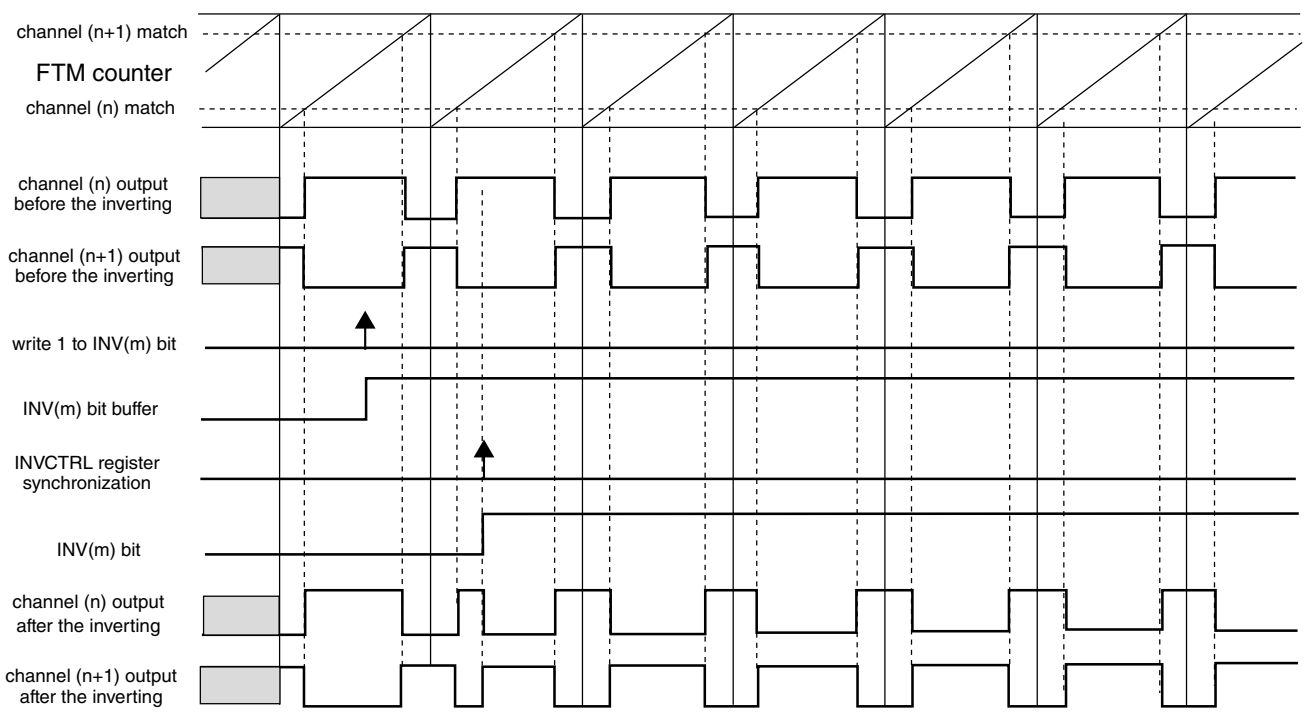
36.4.12 Inverting

The invert functionality swaps the signals between channel (n) and channel (n+1) outputs. The inverting operation is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 1, and
- INVm = 1 (where m represents a channel pair)

The INVm bit in INVCTRL register is updated with its buffer value according to INVCTRL register synchronization

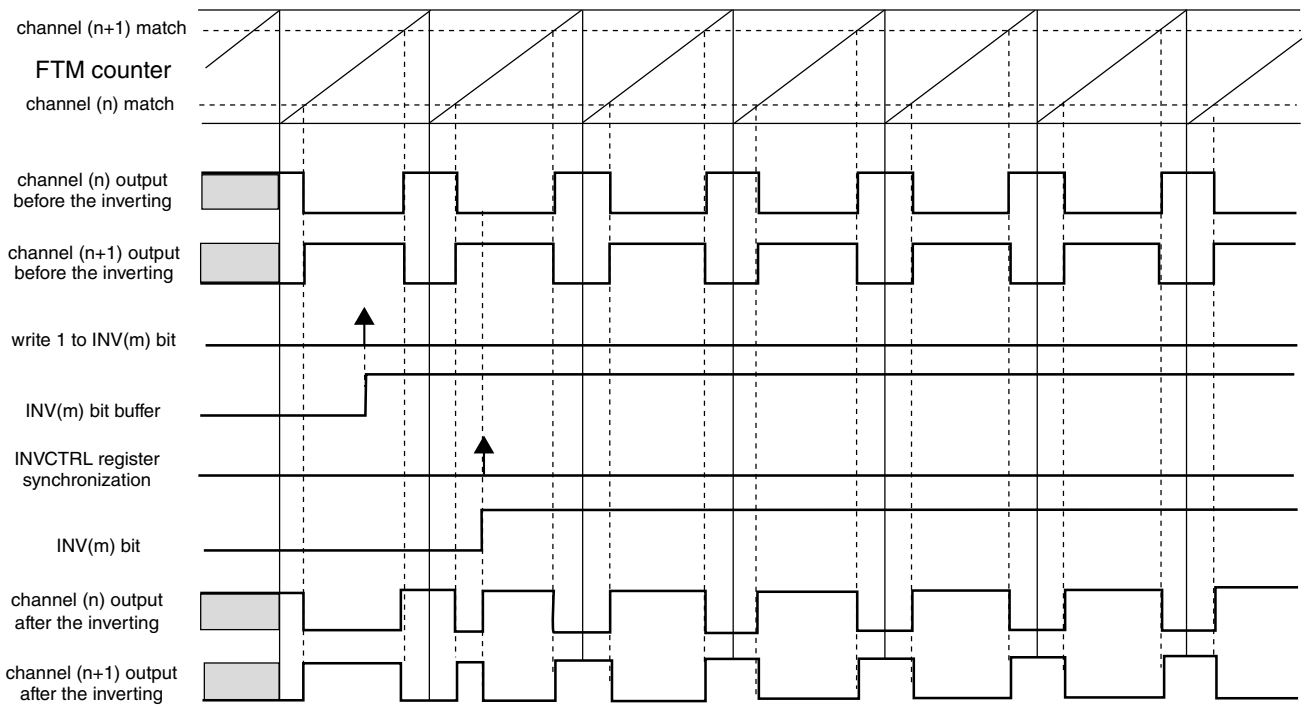
In High-True (ELSnB:ELSnA = 1:0) Combine mode, the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN), forced high at the channel (n) match and forced low at the channel (n+1) match. If the inverting is selected, the channel (n) output behavior is changed to force high at the beginning of the PWM period, force low at the channel (n) match and force high at the channel (n+1) match. See the following figure.



NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

Figure 36-63. Channels (n) and (n+1) outputs after the inverting in High-True (ELSnB:ELSnA = 1:0) Combine mode

Note that the ELSnB:ELSnA bits value should be considered because they define the active state of the channels outputs. In Low-True (ELSnB:ELSnA = X:1) Combine mode, the channel (n) output is forced high at the beginning of the period, forced low at the channel (n) match and forced high at the channel (n+1) match. When inverting is selected, the channels (n) and (n+1) present waveforms as shown in the following figure.



NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

Figure 36-64. Channels (n) and (n+1) outputs after the inverting in Low-True (ELSnB:ELSnA = X:1) Combine mode

Note

The inverting feature is not available in Output Compare mode.

36.4.13 Software output control

The software output control forces the channel output according to software defined values at a specific time in the PWM generation.

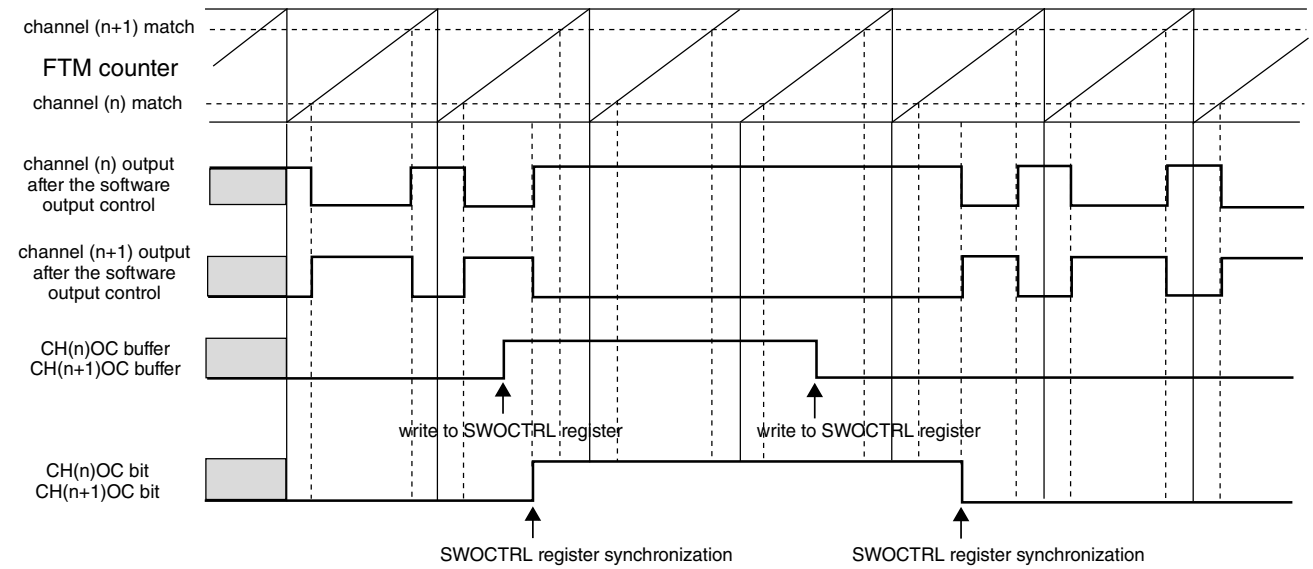
The software output control is selected when:

- QUADEN = 0
- DECAPEN = 0, and
- CHnOC = 1

The CHnOC bit enables the software output control for a specific channel output and the CHnOCV selects the value that is forced to this channel output.

Both CHnOC and CHnOCV bits in SWOCTRL register are buffered and updated with their buffer value according to SWOCTRL register synchronization.

The following figure shows the channels (n) and (n+1) outputs signals when the software output control is used. In this case the channels (n) and (n+1) are set to Combine and Complementary mode.



NOTE
CH(n)OCV = 1 and CH(n+1)OCV = 0.

Figure 36-65. Example of software output control in Combine and Complementary mode

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is zero.

Table 36-8. Software ouput control behavior when (COMP = 0)

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	Channel (n) Output	Channel (n+1) Output
0	0	X	X	is not modified by SWOC	is not modified by SWOC
1	1	0	0	is forced to zero	is forced to zero
1	1	0	1	is forced to zero	is forced to one
1	1	1	0	is forced to one	is forced to zero
1	1	1	1	is forced to one	is forced to one

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is one.

Table 36-9. Software ouput control behavior when (COMP = 1)

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	Channel (n) Output	Channel (n+1) Output
0	0	X	X	is not modified by SWOC	is not modified by SWOC
1	1	0	0	is forced to zero	is forced to zero
1	1	0	1	is forced to zero	is forced to one
1	1	1	0	is forced to one	is forced to zero
1	1	1	1	is forced to one	is forced to zero

Note

- The CH(n)OC and CH(n+1)OC bits should be equal.
- The COMP bit must not be modified when software output control is enabled, that is, CH(n)OC = 1 and/or CH(n +1)OC = 1.
- Software output control has the same behavior with disabled or enabled FTM counter (see the CLKS field description in the Status and Control register).

36.4.14 Deadtime insertion

The deadtime insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non- zero).

DEADTIME register defines the deadtime delay that can be used for all FTM channels. The DTPS[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the deadtime modulo, that is, the number of the deadtime prescaler clocks.

The deadtime delay insertion ensures that no two complementary signals (channels (n) and (n+1)) drive the active state at the same time.

If POL(n) = 0, POL(n+1) = 0, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the low value until the end of the deadtime delay when the channel (n) output is set. Similarly, when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the low value until the end of the deadtime delay when the channel (n+1) output is set. See the following figures.

If POL(n) = 1, POL(n+1) = 1, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the high value until the end of the deadtime delay when the channel (n) output is cleared. Similarly,

when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the high value until the end of the deadtime delay when the channel (n +1) output is cleared.

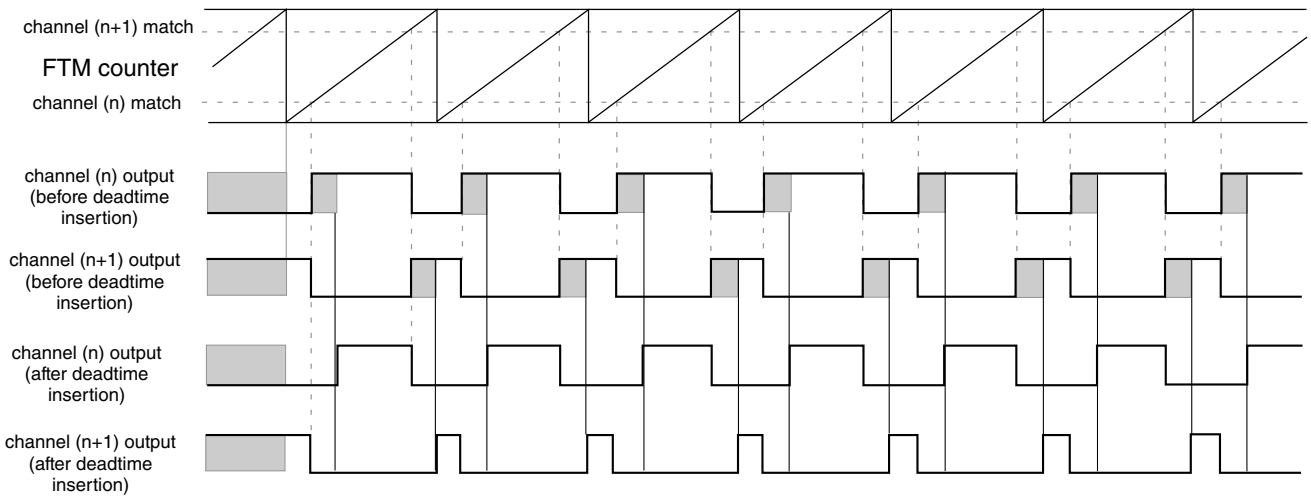


Figure 36-66. Deadtime insertion with ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0

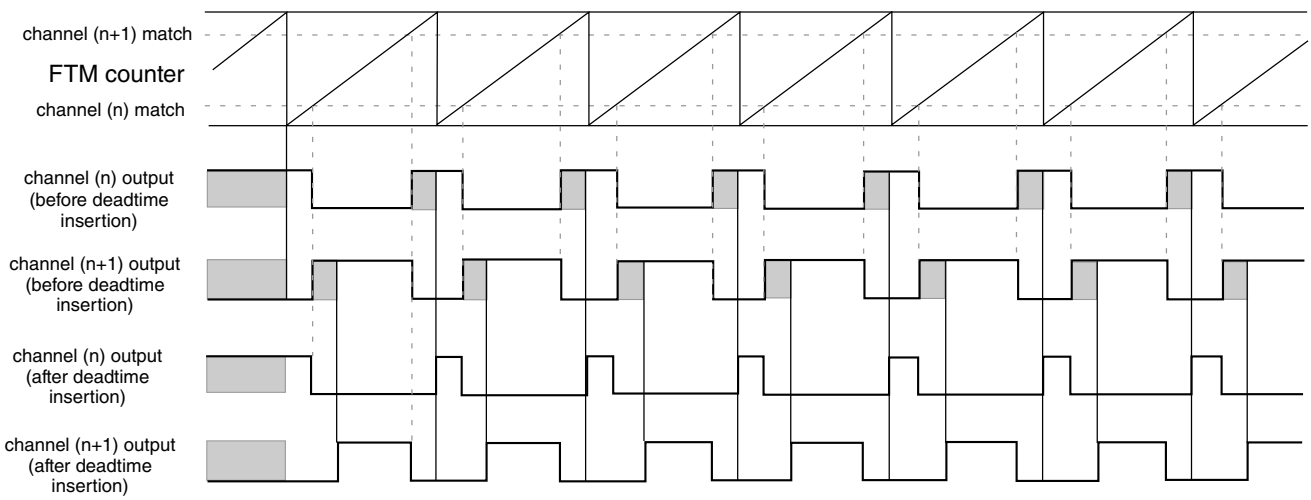


Figure 36-67. Deadtime insertion with ELSnB:ELSnA = X:1, POL(n) = 0, and POL(n+1) = 0

NOTE

- The deadtime feature must be used only in Complementary mode.
- The deadtime feature is not available in Output Compare mode.

36.4.14.1 Deadtime insertion corner cases

If (PS[2:0] is cleared), (DTPS[1:0] = 0:0 or DTPS[1:0] = 0:1):

- and the deadtime delay is greater than or equal to the channel (n) duty cycle ((C(n +1)V – C(n)V) × system clock), then the channel (n) output is always the inactive value (POL(n) bit value).
- and the deadtime delay is greater than or equal to the channel (n+1) duty cycle ((MOD – CNTIN + 1 – (C(n+1)V – C(n)V)) × system clock), then the channel (n+1) output is always the inactive value (POL(n+1) bit value).

Although, in most cases the deadtime delay is not comparable to channels (n) and (n+1) duty cycle, the following figures show examples where the deadtime delay is comparable to the duty cycle.

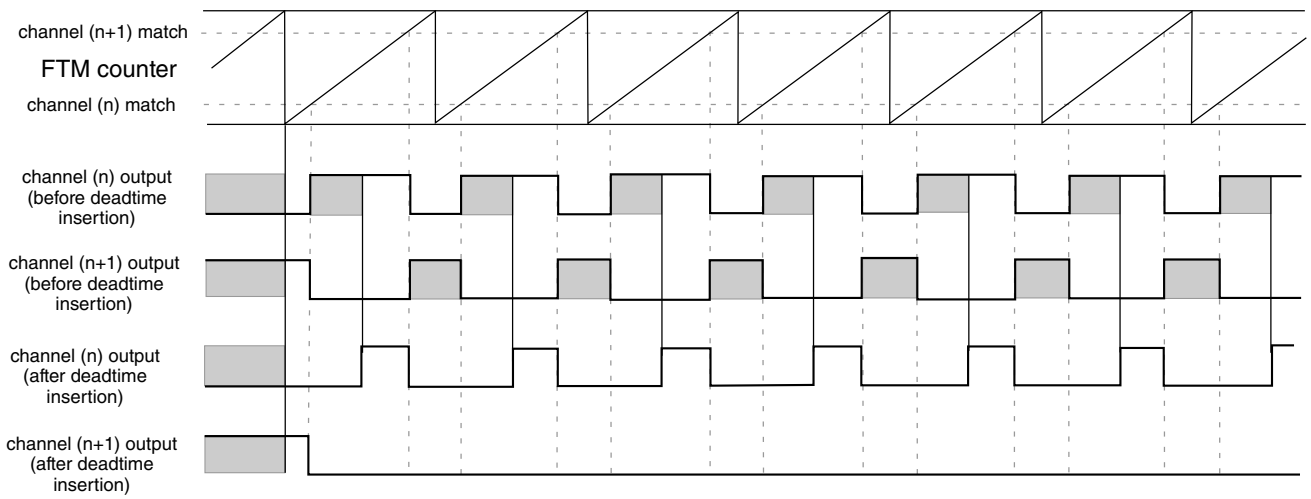


Figure 36-68. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channel (n+1) duty cycle

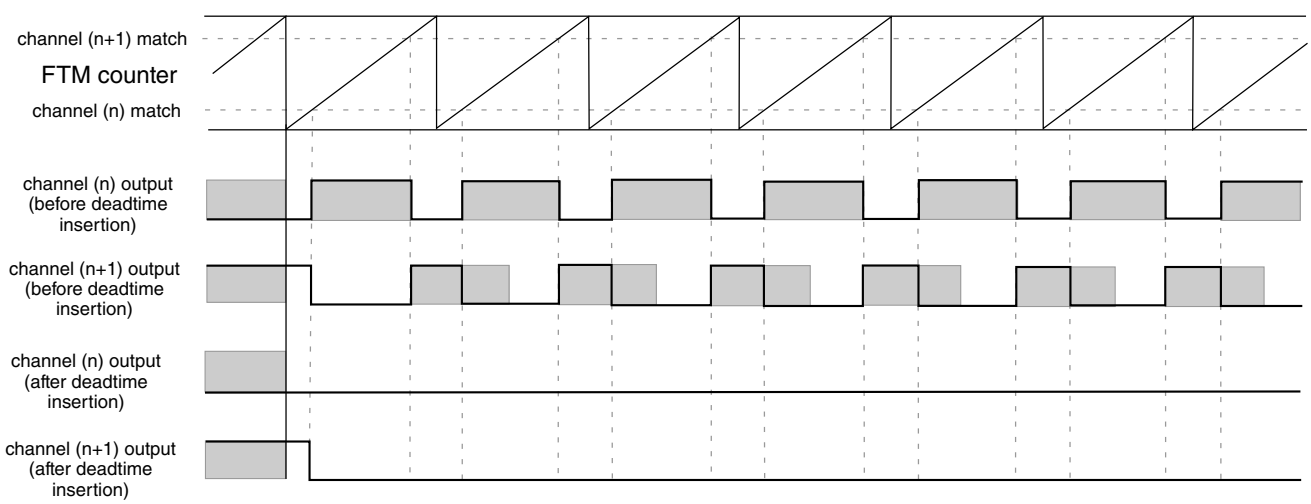


Figure 36-69. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channels (n) and (n+1) duty cycle

36.4.15 Output mask

The output mask can be used to force channels output to their inactive state through software. For example: to control a BLDC motor.

Any write to the OUTMASK register updates its write buffer. The OUTMASK register is updated with its buffer value by PWM synchronization; see OUTMASK register synchronization.

If CHnOM = 1, then the channel (n) output is forced to its inactive state (POLn bit value). If CHnOM = 0, then the channel (n) output is unaffected by the output mask. See the following figure.

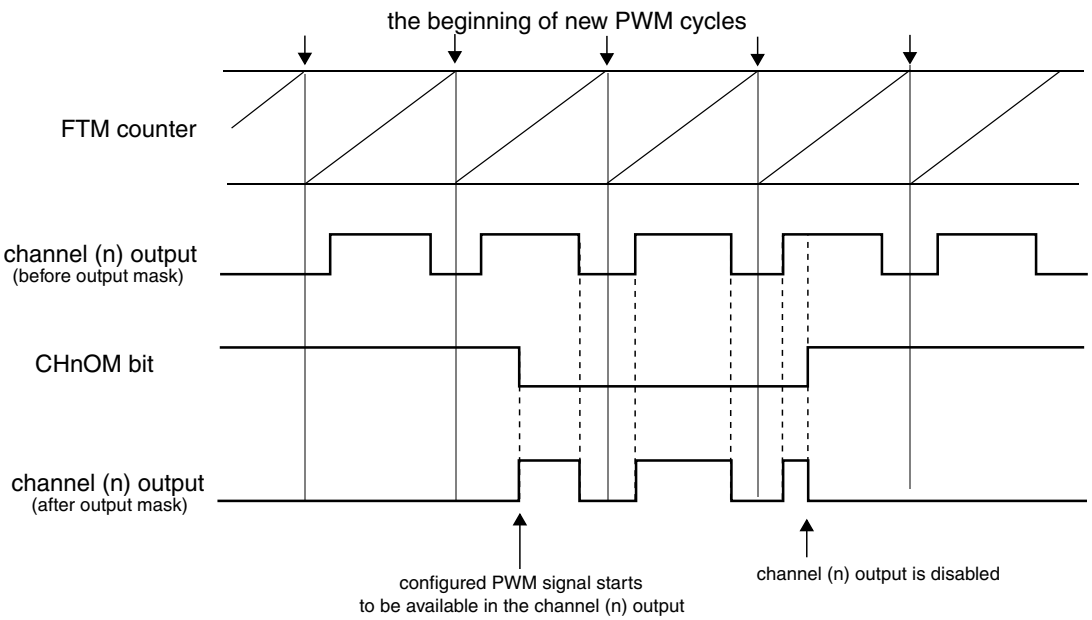


Figure 36-70. Output mask with POLn = 0

The following table shows the output mask result before the polarity control.

Table 36-10. Output mask result for channel (n) before the polarity control

CHnOM	Output Mask Input	Output Mask Result
0	inactive state	inactive state
	active state	active state
1	inactive state	inactive state
	active state	

36.4.16 Fault control

The fault control is enabled if (FAULTM[1:0] ≠ 0:0).

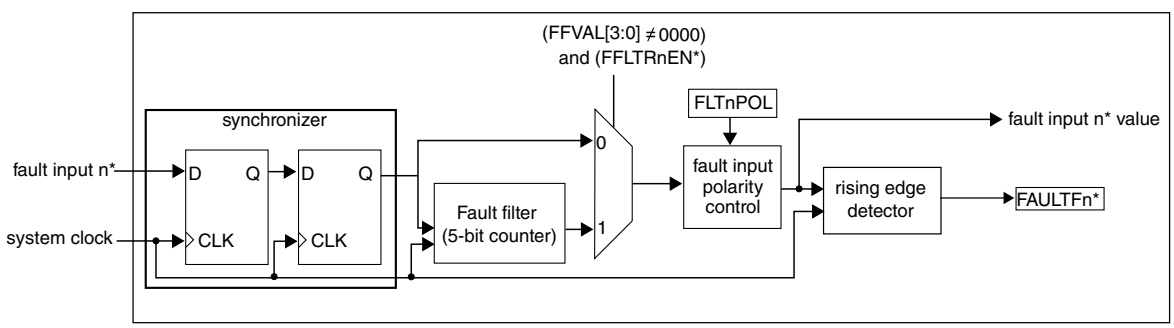
FTM can have up to four fault inputs. FAULTnEN bit (where n = 0, 1, 2, 3) enables the fault input n and FFLTRnEN bit enables the fault input n filter. FFVAL[3:0] bits select the value of the enabled filter in each enabled fault input.

First, each fault input signal is synchronized by the system clock; see the synchronizer block in the following figure. Following synchronization, the fault input n signal enters the filter block. When there is a state change in the fault input n signal, the 5-bit counter is reset and starts counting up. As long as the new state is stable on the fault input n, the counter continues to increment. If the 5-bit counter overflows, that is, the counter exceeds the value of the FFVAL[3:0] bits, the new fault input n value is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the fault input n signal before validation (counter overflow), the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by FFVAL[3:0] bits (× system clock) is regarded as a glitch and is not passed on to the edge detector.

The fault input n filter is disabled when the FFVAL[3:0] bits are zero or when FAULTnEN = 0. In this case, the fault input n signal is delayed 2 rising edges of the system clock and the FAULTFn bit is set on 3th rising edge of the system clock after a rising edge occurs on the fault input n.

If FFVAL[3:0] ≠ 0000 and FAULTnEN = 1, then the fault input n signal is delayed (3 + FFVAL[3:0]) rising edges of the system clock, that is, the FAULTFn bit is set (4 + FFVAL[3:0]) rising edges of the system clock after a rising edge occurs on the fault input n.



* where n = 3, 2, 1, 0

Figure 36-71. Fault input n control block diagram

If the fault control and fault input n are enabled and a rising edge at the fault input n signal is detected, a fault condition has occurred and the FAULTFn bit is set. The FAULTF bit is the logic OR of FAULTFn[3:0] bits. See the following figure.

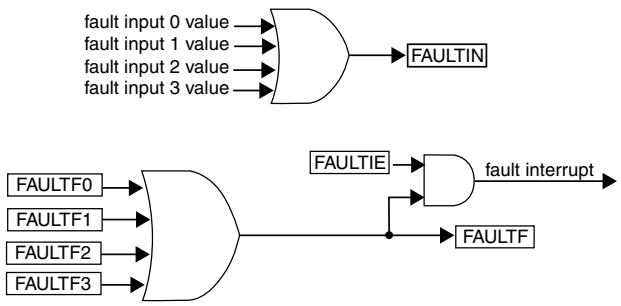


Figure 36-72. FAULTF and FAULTIN bits and fault interrupt

If the fault control is enabled (FAULTM[1:0] ≠ 0:0), a fault condition has occurred and (FAULTEN = 1), then outputs are forced to their safe values:

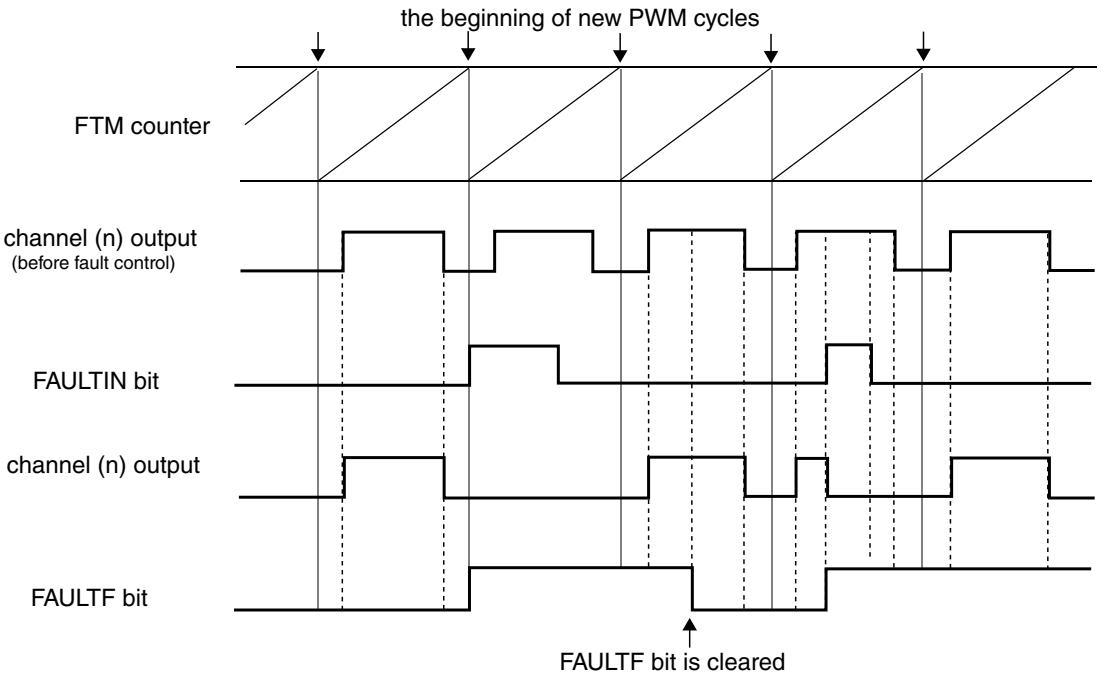
- Channel (n) output takes the value of POL(n)
- Channel (n+1) takes the value of POL(n+1)

The fault interrupt is generated when (FAULTF = 1) and (FAULTIE = 1). This interrupt request remains set until:

- Software clears the FAULTF bit by reading FAULTF bit as 1 and writing 0 to it
- Software clears the FAULTIE bit
- A reset occurs

36.4.16.1 Automatic fault clearing

If the automatic fault clearing is selected (FAULTM[1:0] = 1:1), then the channels output disabled by fault control is again enabled when the fault input signal (FAULTIN) returns to zero and a new PWM cycle begins. See the following figure.

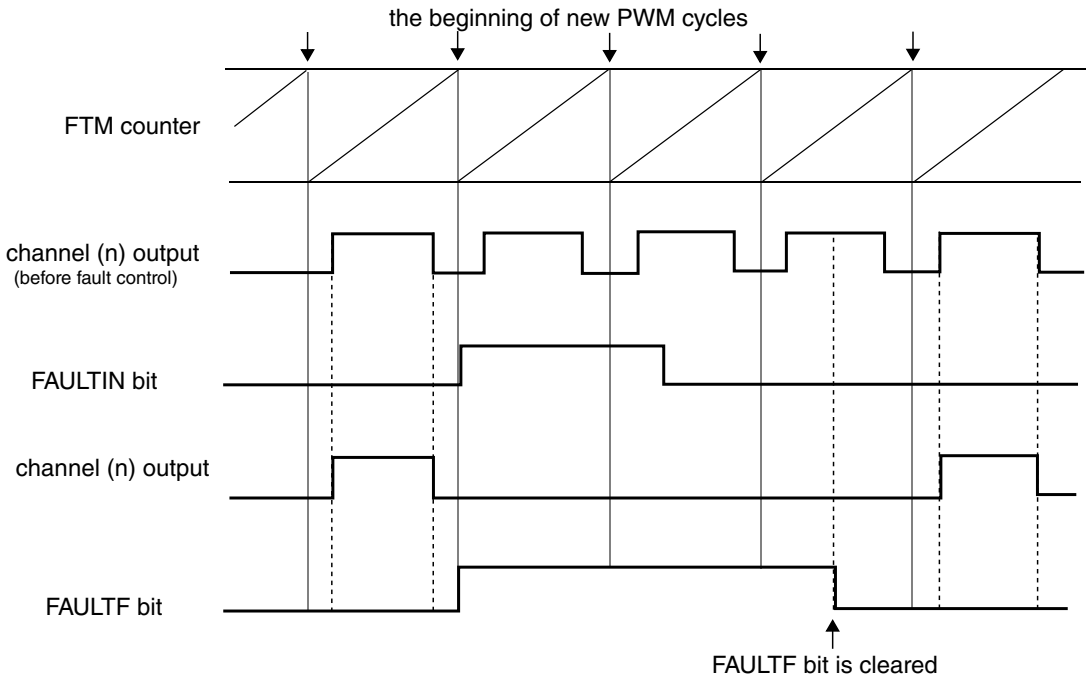


NOTE
The channel (n) output is after the fault control with automatic fault clearing and POLn = 0.

Figure 36-73. Fault control with automatic fault clearing

36.4.16.2 Manual fault clearing

If the manual fault clearing is selected (FAULTM[1:0] = 0:1 or 1:0), then the channels output disabled by fault control is again enabled when the FAULTF bit is cleared and a new PWM cycle begins. See the following figure.



NOTE
The channel (n) output is after the fault control with manual fault clearing and POLn = 0.

Figure 36-74. Fault control with manual fault clearing

36.4.16.3 Fault inputs polarity control

The FLTjPOL bit selects the fault input j polarity, where j = 0, 1, 2, 3:

- If FLTjPOL = 0, the fault j input polarity is high, so the logical one at the fault input j indicates a fault.
- If FLTjPOL = 1, the fault j input polarity is low, so the logical zero at the fault input j indicates a fault.

36.4.17 Polarity control

The POLn bit selects the channel (n) output polarity:

- If POLn = 0, the channel (n) output polarity is high, so the logical one is the active state and the logical zero is the inactive state.
- If POLn = 1, the channel (n) output polarity is low, so the logical zero is the active state and the logical one is the inactive state.

36.4.18 Initialization

The initialization forces the CHnOI bit value to the channel (n) output when a one is written to the INIT bit.

The initialization depends on COMP and DTEN bits. The following table shows the values that channels (n) and (n+1) are forced by initialization when the COMP and DTEN bits are zero.

Table 36-11. Initialization behavior when (COMP = 0 and DTEN = 0)

CH(n)OI	CH(n+1)OI	Channel (n) Output	Channel (n+1) Output
0	0	is forced to zero	is forced to zero
0	1	is forced to zero	is forced to one
1	0	is forced to one	is forced to zero
1	1	is forced to one	is forced to one

The following table shows the values that channels (n) and (n+1) are forced by initialization when (COMP = 1) or (DTEN = 1).

Table 36-12. Initialization behavior when (COMP = 1 or DTEN = 1)

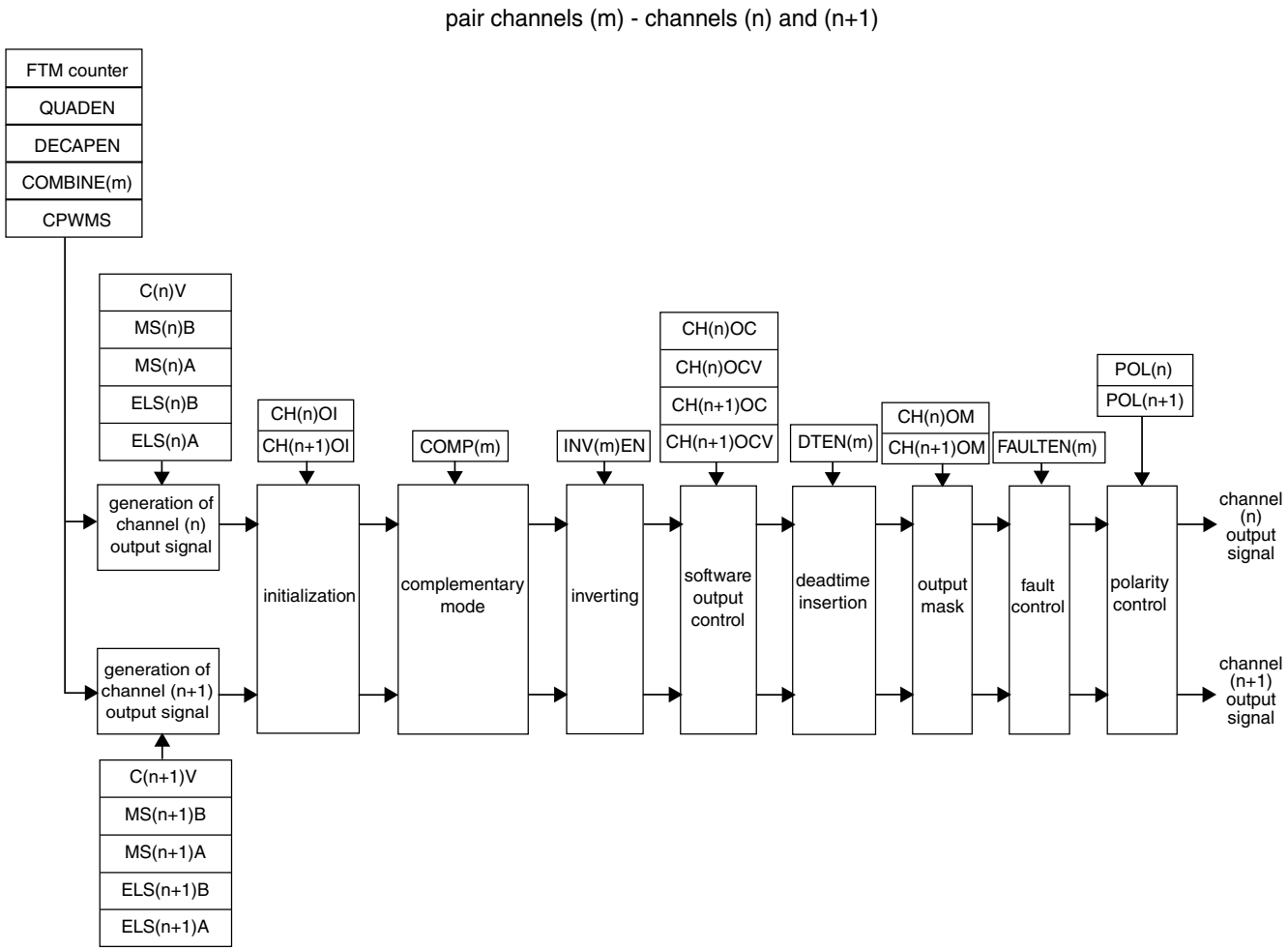
CH(n)OI	CH(n+1)OI	Channel (n) Output	Channel (n+1) Output
0	X	is forced to zero	is forced to one
1	X	is forced to one	is forced to zero

Note

The initialization feature must be used only with disabled FTM counter. See the description of the CLKS field in the Status and Control register.

36.4.19 Features priority

The following figure shows the priority of the features used at the generation of channels (n) and (n+1) outputs signals.



NOTE

The channels (n) and (n+1) are in output compare, EPWM, CPWM or combine modes.

Figure 36-75. Priority of the features used at the generation of channels (n) and (n+1) outputs signals

Note

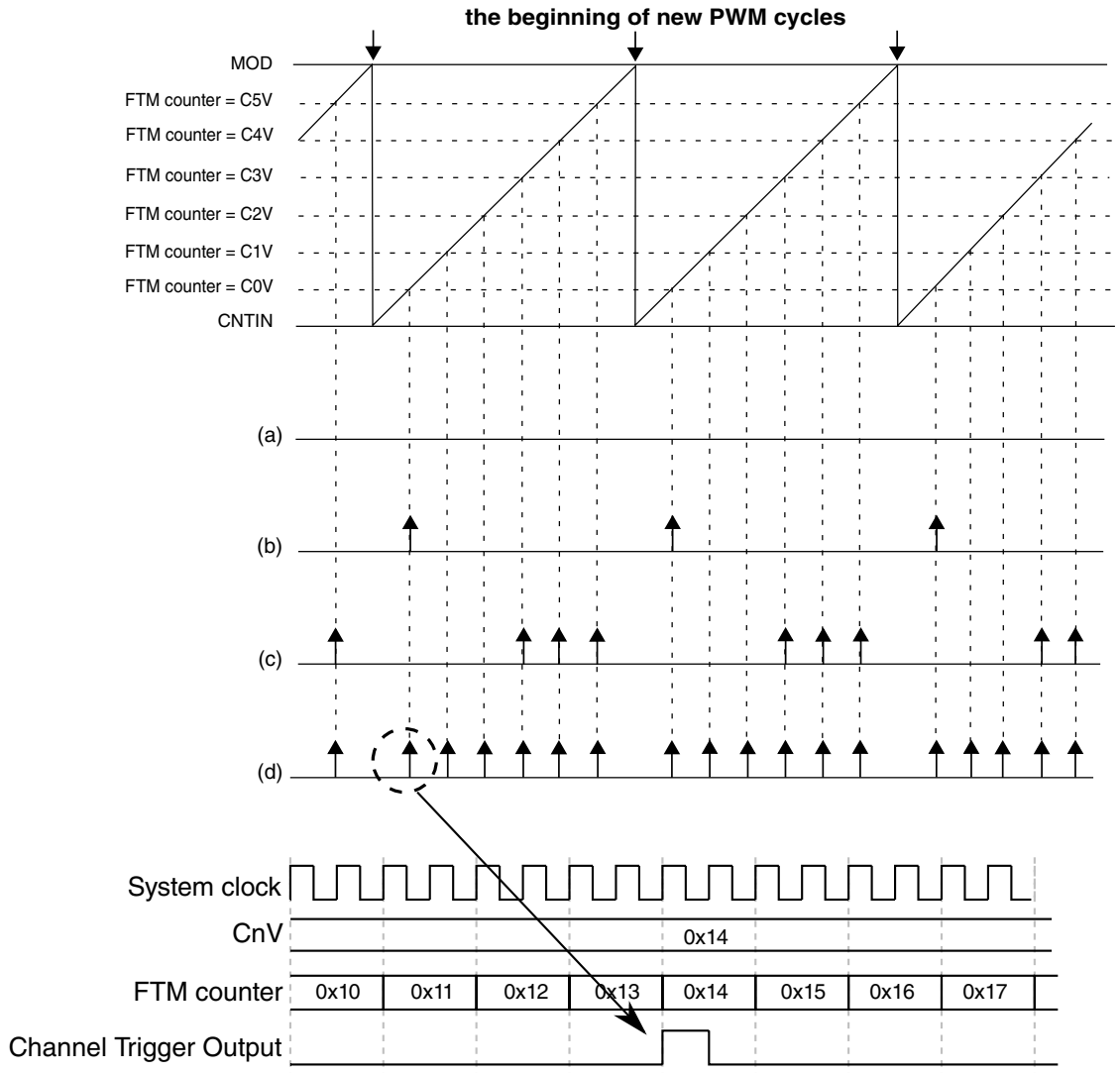
The Initialization feature must not be used with Inverting and Software output control features.

36.4.20 Channel trigger output

If CH(j)TRIG bit of the FTM External Trigger (FTM_EXTTRIG) register is set, where j = 0, 1, 2, 3, 4, or 5, then the FTM generates a trigger when the channel (j) match occurs (FTM counter = C(j)V).

The channel trigger output provides a trigger signal which has one FTM clock period width and is used for on-chip modules.

The FTM is able to generate multiple triggers in one PWM period. Because each trigger is generated for a specific channel, several channels are required to implement this functionality. This behavior is described in the following figure.



NOTE

(a) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(b) CH0TRIG = 1, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(c) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1
(d) CH0TRIG = 1, CH1TRIG = 1, CH2TRIG = 1, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1

Figure 36-76. Channel match trigger

36.4.21 Initialization trigger

If INITTRIGEN = 1, then the FTM generates a trigger when the FTM counter is updated with the CNTIN register value in the following cases.

- The FTM counter is automatically updated with the CNTIN register value by the selected counting mode.
- When there is a write to CNT register.
- When there is the FTM counter synchronization.
- If (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits.
- If the channel (n) is in Input Capture mode, (ICRST = 1) and the selected input capture event occurs in the channel (n) input.

The following figures show these cases.

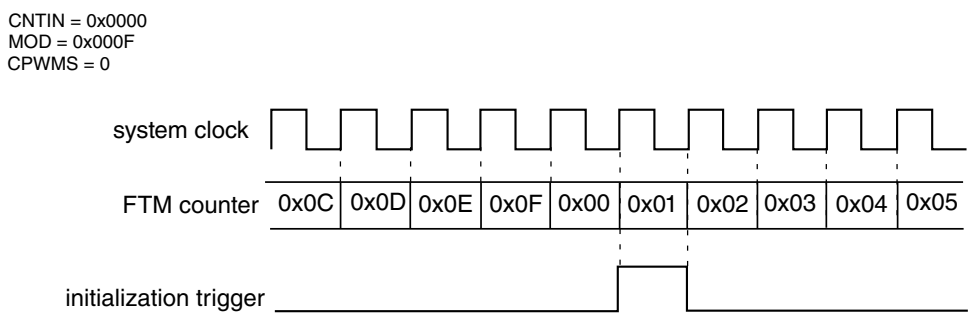


Figure 36-77. Initialization trigger is generated when the FTM counting achieves the CNTIN register value

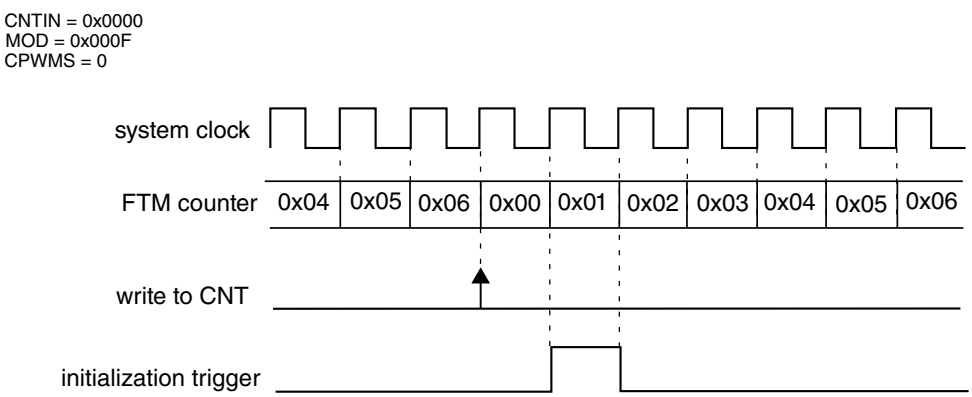


Figure 36-78. Initialization trigger is generated when there is a write to CNT register

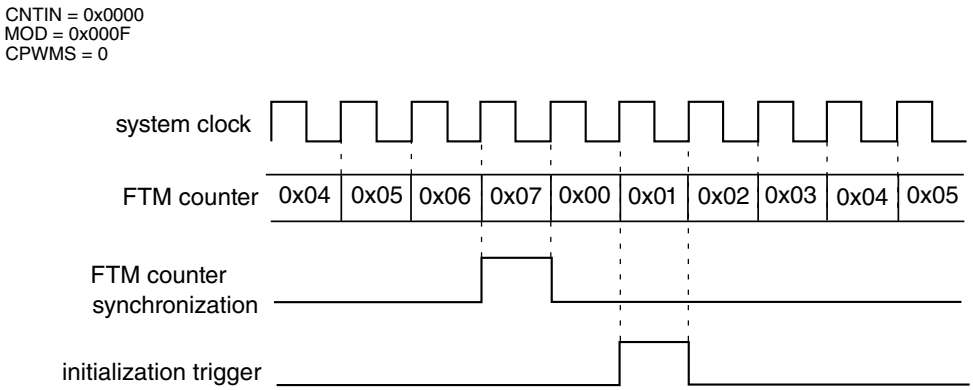


Figure 36-79. Initialization trigger is generated when there is the FTM counter synchronization

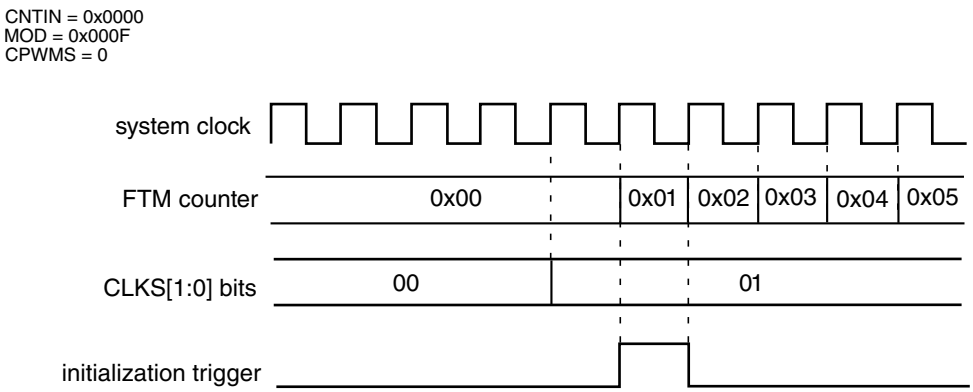
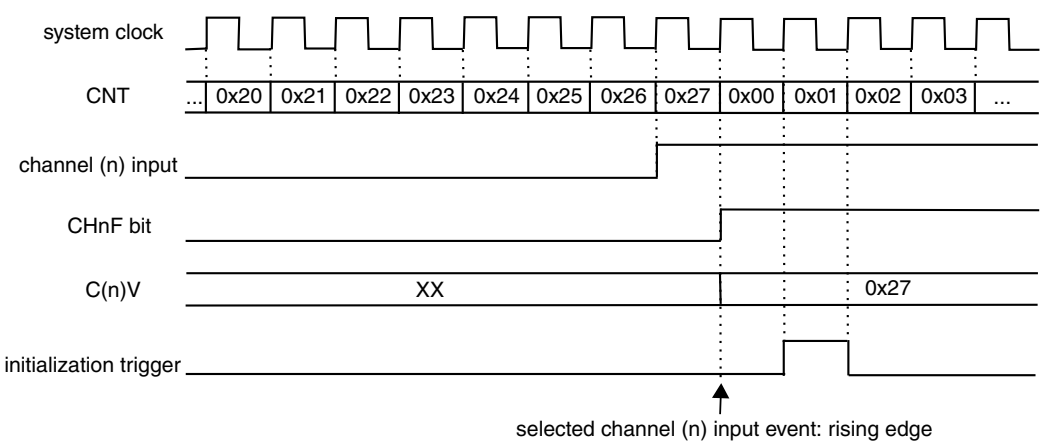


Figure 36-80. Initialization trigger is generated if (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits



NOTE
Channel (n) input after its synchronizer and filter
MOD = 0xFFFF
CNTIN = 0x0000
PS[2:0] = 3'b000
ICRST = 1'b1

Figure 36-81. Initialization trigger is generated if the channel (n) is in Input Capture mode, ICRST = 1 and the selected input capture event occurs in the channel (n) input

The initialization trigger output provides a trigger signal that is used for on-chip modules.

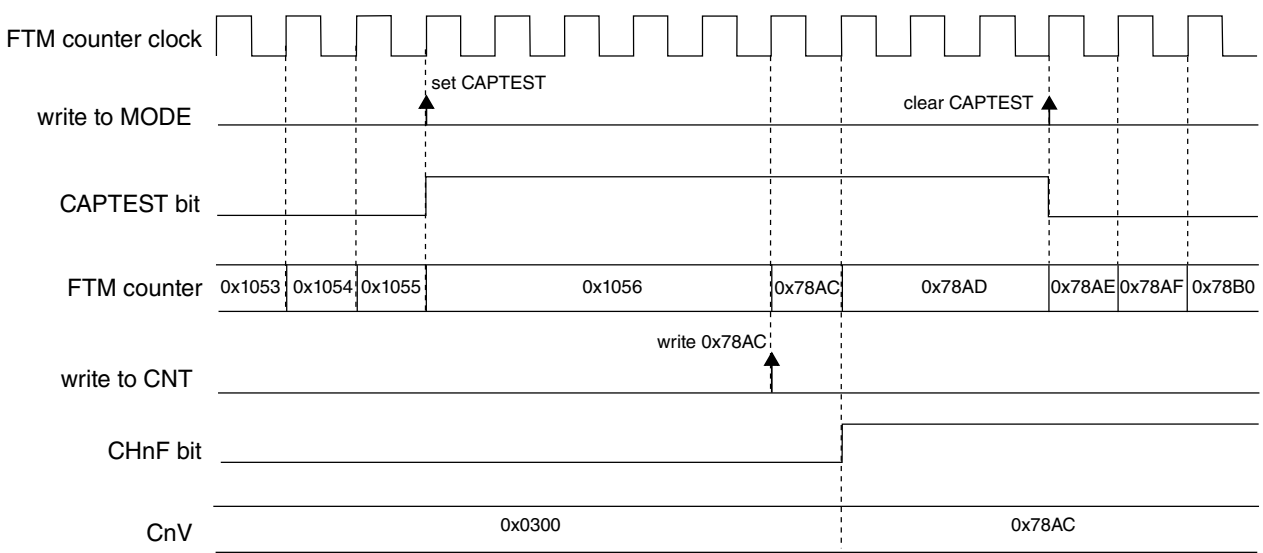
36.4.22 Capture Test mode

The Capture Test mode allows to test the CnV registers, the FTM counter and the interconnection logic between the FTM counter and CnV registers.

In this test mode, all channels must be configured for Input Capture mode and FTM counter must be configured to the Up counting.

When the Capture Test mode is enabled (CAPTEST = 1), the FTM counter is frozen and any write to CNT register updates directly the FTM counter; see the following figure. After it was written, all CnV registers are updated with the written value to CNT register and CHnF bits are set. Therefore, the FTM counter is updated with its next value according to its configuration. Its next value depends on CNTIN, MOD, and the written value to FTM counter.

The next reads of CnV registers return the written value to the FTM counter and the next reads of CNT register return FTM counter next value.



NOTE
- FTM counter configuration: (FTMEN = 1), (QUADEN = 0), (CAPTEST = 1), (CPWMS = 0), (CNTIN = 0x0000), and (MOD = 0xFFFF)
- FTM channel n configuration: input capture mode - (DECAPEN = 0), (COMBINE = 0), and (MSnB:MSnA = 0:0)

Figure 36-82. Capture Test mode

36.4.23 DMA

The channel generates a DMA transfer request according to DMA and CHnIE bits. See the following table.

Table 36-13. Channel DMA transfer request

DMA	CHnIE	Channel DMA Transfer Request	Channel Interrupt
0	0	The channel DMA transfer request is not generated.	The channel interrupt is not generated.
0	1	The channel DMA transfer request is not generated.	The channel interrupt is generated if (CHnF = 1).
1	0	The channel DMA transfer request is not generated.	The channel interrupt is not generated.
1	1	The channel DMA transfer request is generated if (CHnF = 1).	The channel interrupt is not generated.

If DMA = 1, the CHnF bit is cleared either by channel DMA transfer done or reading CnSC while CHnF is set and then writing a zero to CHnF bit according to CHnIE bit. See the following table.

Table 36-14. Clear CHnF bit when DMA = 1

CHnIE	How CHnF Bit Can Be Cleared
0	CHnF bit is cleared either when the channel DMA transfer is done or by reading CnSC while CHnF is set and then writing a 0 to CHnF bit.
1	CHnF bit is cleared when the channel DMA transfer is done.

36.4.24 Dual Edge Capture mode

The Dual Edge Capture mode is selected if DECAPEN = 1. This mode allows to measure a pulse width or period of the signal on the input of channel (n) of a channel pair. The channel (n) filter can be active in this mode when n is 0 or 2.

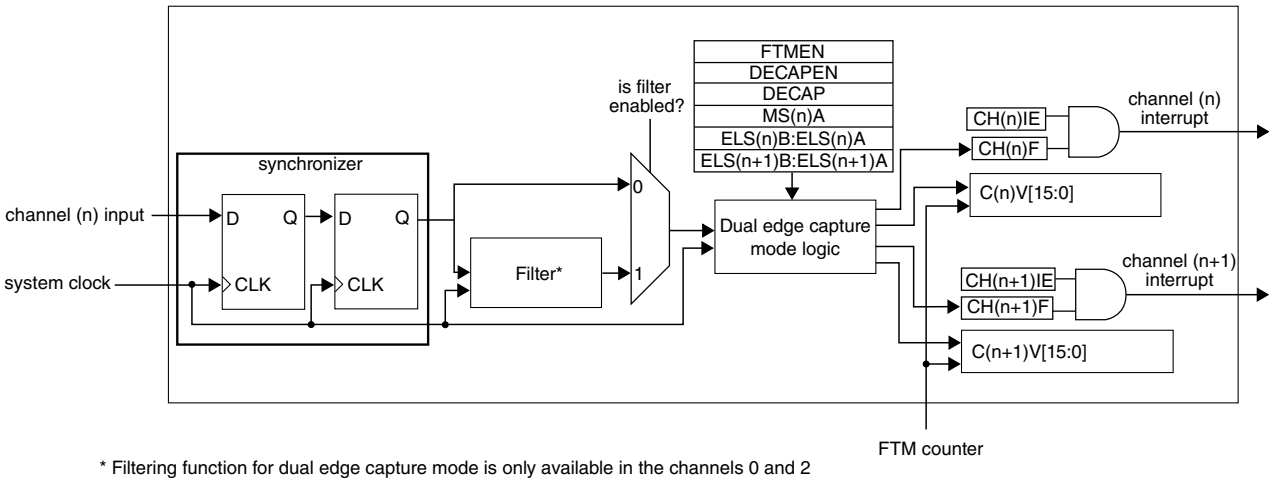


Figure 36-83. Dual Edge Capture mode block diagram

The MS(n)A bit defines if the Dual Edge Capture mode is one-shot or continuous.

The ELS(n)B:ELS(n)A bits select the edge that is captured by channel (n), and ELS(n +1)B:ELS(n+1)A bits select the edge that is captured by channel (n+1). If both ELS(n)B:ELS(n)A and ELS(n+1)B:ELS(n+1)A bits select the same edge, then it is the period measurement. If these bits select different edges, then it is a pulse width measurement.

In the Dual Edge Capture mode, only channel (n) input is used and channel (n+1) input is ignored.

If the selected edge by channel (n) bits is detected at channel (n) input, then CH(n)F bit is set and the channel (n) interrupt is generated (if CH(n)IE = 1). If the selected edge by channel (n+1) bits is detected at channel (n) input and (CH(n)F = 1), then CH(n+1)F bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1).

The C(n)V register stores the value of FTM counter when the selected edge by channel (n) is detected at channel (n) input. The C(n+1)V register stores the value of FTM counter when the selected edge by channel (n+1) is detected at channel (n) input.

In this mode, a coherency mechanism ensures coherent data when the C(n)V and C(n +1)V registers are read. The only requirement is that C(n)V must be read before C(n +1)V.

Note

- The CH(n)F, CH(n)IE, MS(n)A, ELS(n)B, and ELS(n)A bits are channel (n) bits.

- The CH(n+1)F, CH(n+1)IE, MS(n+1)A, ELS(n+1)B, and ELS(n+1)A bits are channel (n+1) bits.
- The Dual Edge Capture mode must be used with ELS(n)B:ELS(n)A = 0:1 or 1:0, ELS(n+1)B:ELS(n+1)A = 0:1 or 1:0 and the FTM counter in Free running counter.

36.4.24.1 One-Shot Capture mode

The One-Shot Capture mode is selected when (DECAPEN = 1), and (MS(n)A = 0). In this capture mode, only one pair of edges at the channel (n) input is captured. The ELS(n)B:ELS(n)A bits select the first edge to be captured, and ELS(n+1)B:ELS(n+1)A bits select the second edge to be captured.

The edge captures are enabled while DECAP bit is set. For each new measurement in One-Shot Capture mode, first the CH(n)F and CH(n+1) bits must be cleared, and then the DECAP bit must be set.

In this mode, the DECAP bit is automatically cleared by FTM when the edge selected by channel (n+1) is captured. Therefore, while DECAP bit is set, the one-shot capture is in process. When this bit is cleared, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

Similarly, when the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

36.4.24.2 Continuous Capture mode

The Continuous Capture mode is selected when (DECAPEN = 1), and (MS(n)A = 1). In this capture mode, the edges at the channel (n) input are captured continuously. The ELS(n)B:ELS(n)A bits select the initial edge to be captured, and ELS(n+1)B:ELS(n+1)A bits select the final edge to be captured.

The edge captures are enabled while DECAP bit is set. For the initial use, first the CH(n)F and CH(n+1)F bits must be cleared, and then DECAP bit must be set to start the continuous measurements.

When the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers. The latest captured values are always available in these registers even after the DECAP bit is cleared.

In this mode, it is possible to clear only the CH(n+1)F bit. Therefore, when the CH(n+1)F bit is set again, the latest captured values are available in C(n)V and C(n+1)V registers.

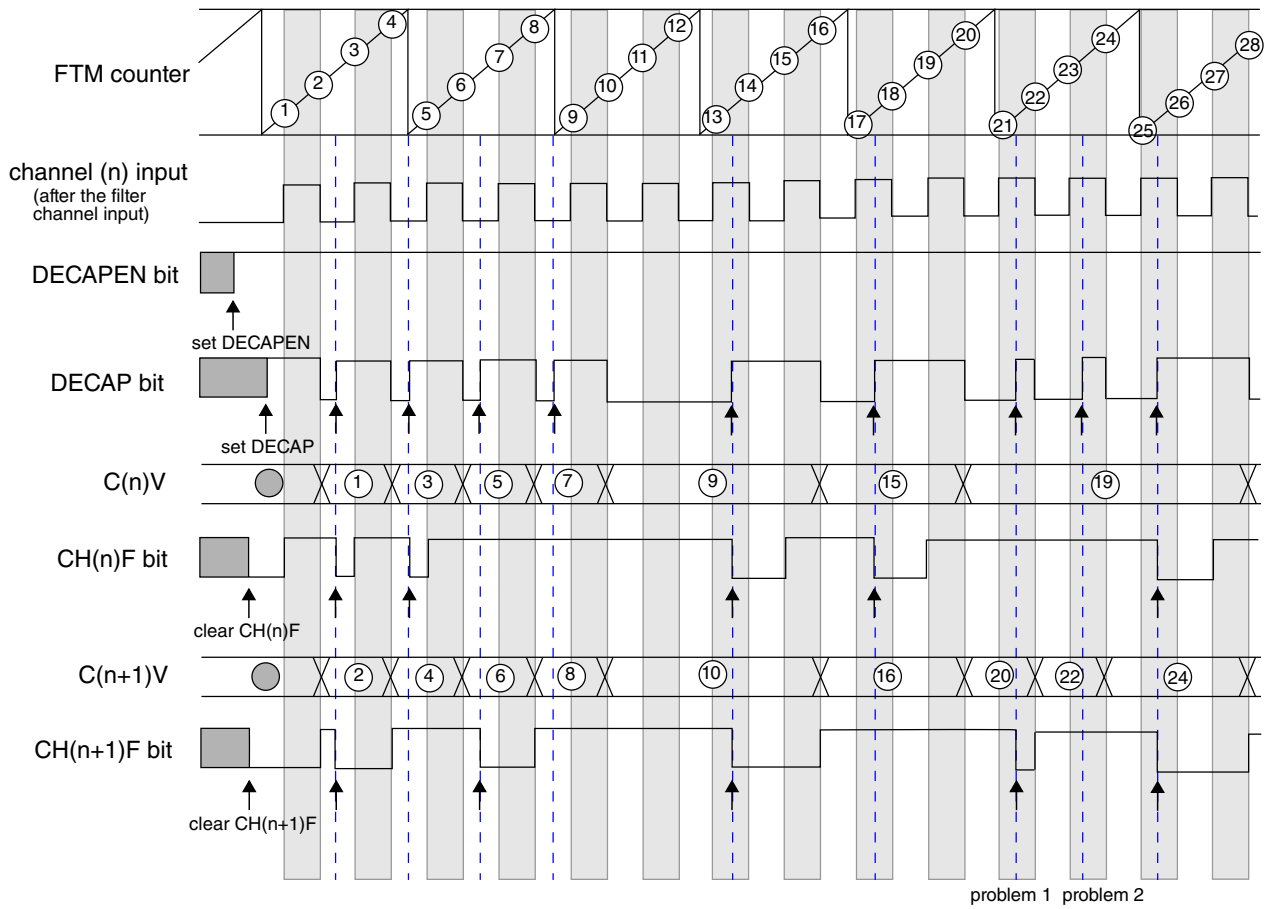
For a new sequence of the measurements in the Dual Edge Capture – Continuous mode, clear the CH(n)F and CH(n+1)F bits to start new measurements.

36.4.24.3 Pulse width measurement

If the channel (n) is configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1) and the channel (n+1) to capture falling edges (ELS(n+1)B:ELS(n+1)A = 1:0), then the positive polarity pulse width is measured. If the channel (n) is configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0) and the channel (n+1) to capture rising edges (ELS(n +1)B:ELS(n+1)A = 0:1), then the negative polarity pulse width is measured.

The pulse width measurement can be made in One-Shot Capture mode or Continuous Capture mode.

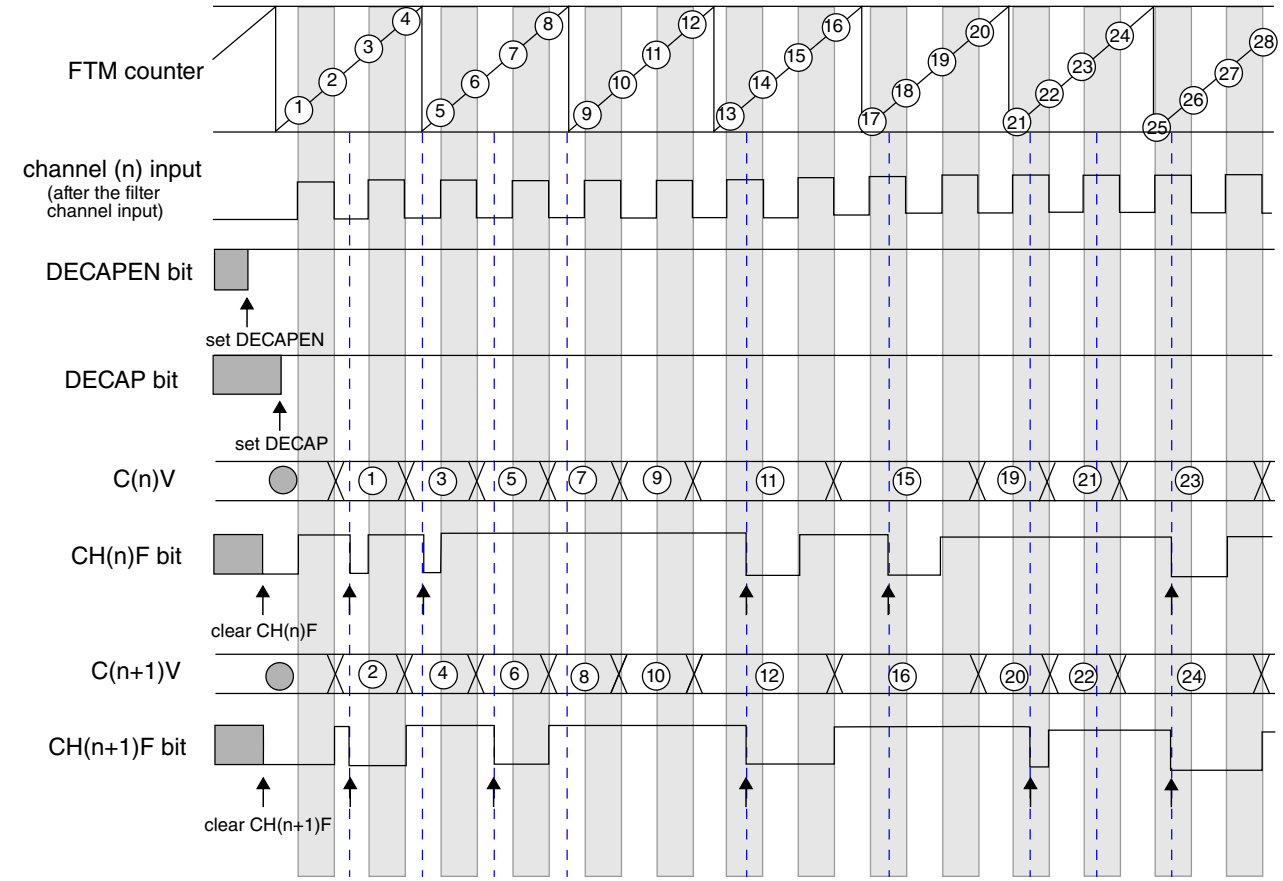
The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the measurement of next positive polarity pulse width. The CH(n)F bit is set when the first edge of this pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second edge of this pulse is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 36-84. Dual Edge Capture – One-Shot mode for positive polarity pulse width measurement

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first edge of the positive polarity pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set when the second edge of this pulse is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. The CH(n+1)F bit indicates when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

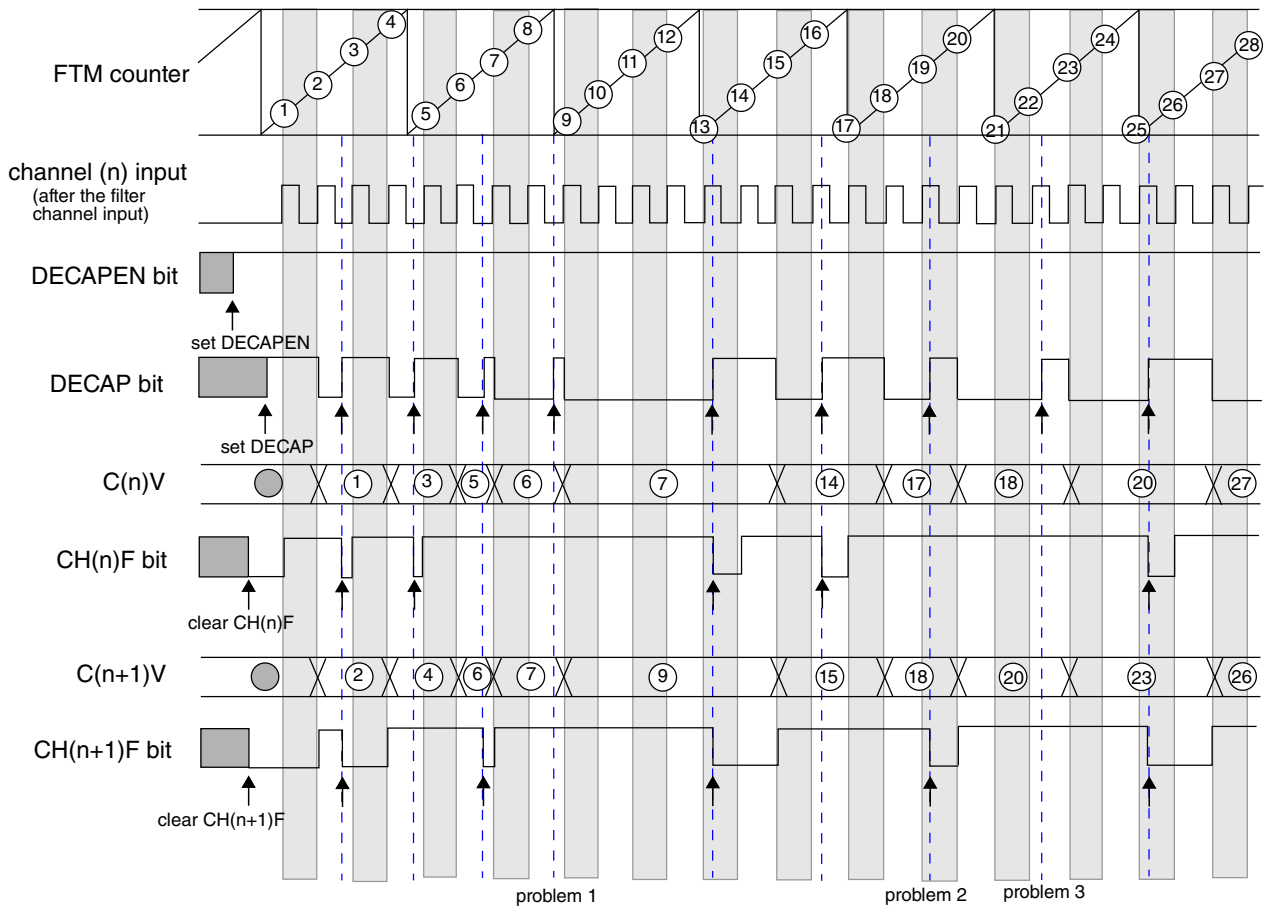
Figure 36-85. Dual Edge Capture – Continuous mode for positive polarity pulse width measurement

36.4.24.4 Period measurement

If the channels (n) and (n+1) are configured to capture consecutive edges of the same polarity, then the period of the channel (n) input signal is measured. If both channels (n) and (n+1) are configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1 and ELS(n +1)B:ELS(n+1)A = 0:1), then the period between two consecutive rising edges is measured. If both channels (n) and (n+1) are configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0 and ELS(n+1)B:ELS(n+1)A = 1:0), then the period between two consecutive falling edges is measured.

The period measurement can be made in One-Shot Capture mode or Continuous Capture mode.

The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the measurement of next period. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second rising edge is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two selected edges were captured and the C(n)V and C(n+1)V registers are ready for reading.

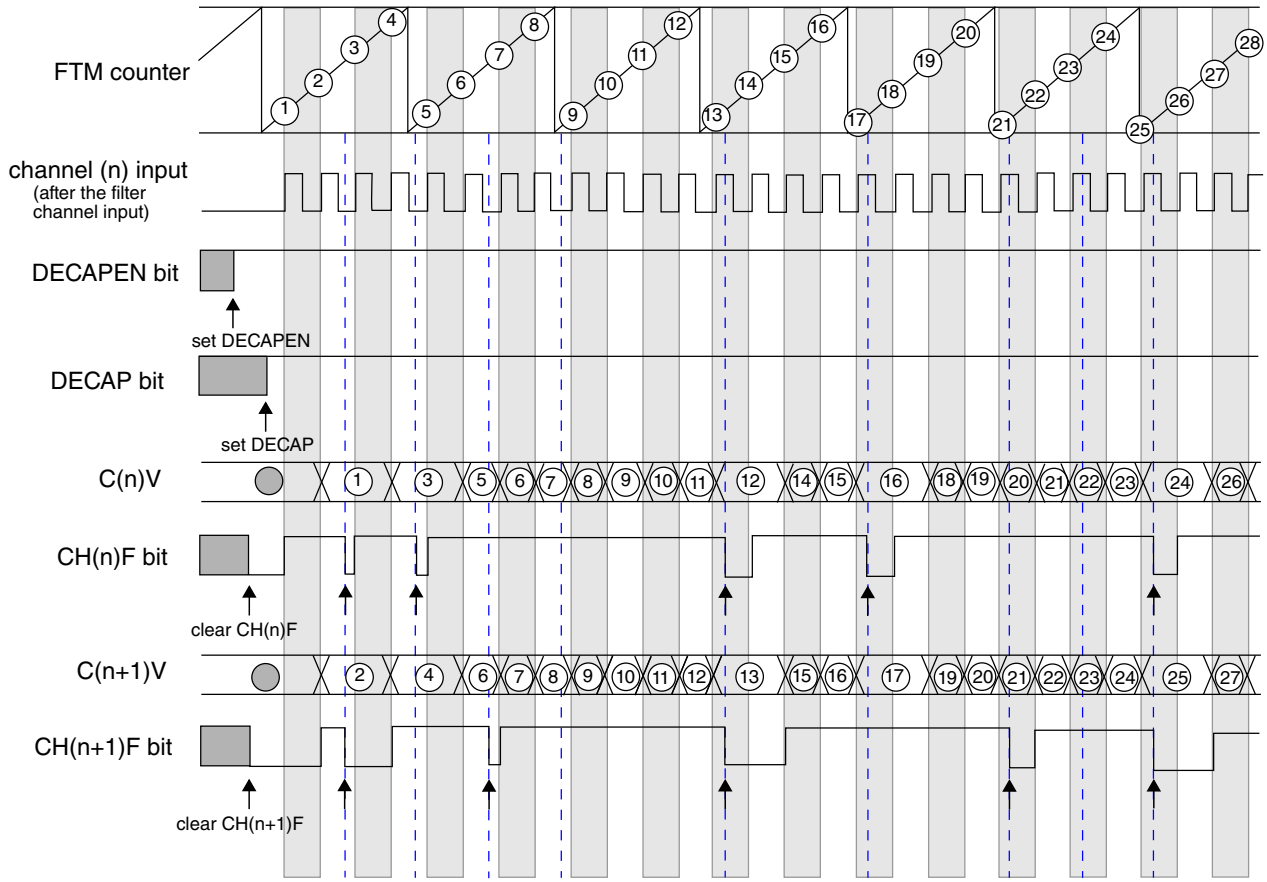


Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 0, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 3: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 36-86. Dual Edge Capture – One-Shot mode to measure of the period between two consecutive rising edges

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set

when the second rising edge is detected, that is, the edge selected by ELS(n+1)B:ELS(n +1)A bits. The CH(n+1)F bit indicates when two edges of the period were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

Figure 36-87. Dual Edge Capture – Continuous mode to measure of the period between two consecutive rising edges

36.4.24.5 Read coherency mechanism

The Dual Edge Capture mode implements a read coherency mechanism between the FTM counter value captured in C(n)V and C(n+1)V registers. The read coherency mechanism is illustrated in the following figure. In this example, the channels (n) and (n +1) are in Dual Edge Capture – Continuous mode for positive polarity pulse width measurement. Thus, the channel (n) is configured to capture the FTM counter value when there is a rising edge at channel (n) input signal, and channel (n+1) to capture the FTM counter value when there is a falling edge at channel (n) input signal.

When a rising edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n) capture buffer. The channel (n) capture buffer value is transferred to C(n)V register when a falling edge occurs in the channel (n) input signal. C(n)V register has the FTM counter value when the previous rising edge occurred, and the channel (n) capture buffer has the FTM counter value when the last rising edge occurred.

When a falling edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n+1) capture buffer. The channel (n+1) capture buffer value is transferred to C(n+1)V register when the C(n)V register is read.

In the following figure, the read of C(n)V returns the FTM counter value when the event 1 occurred and the read of C(n+1)V returns the FTM counter value when the event 2 occurred.

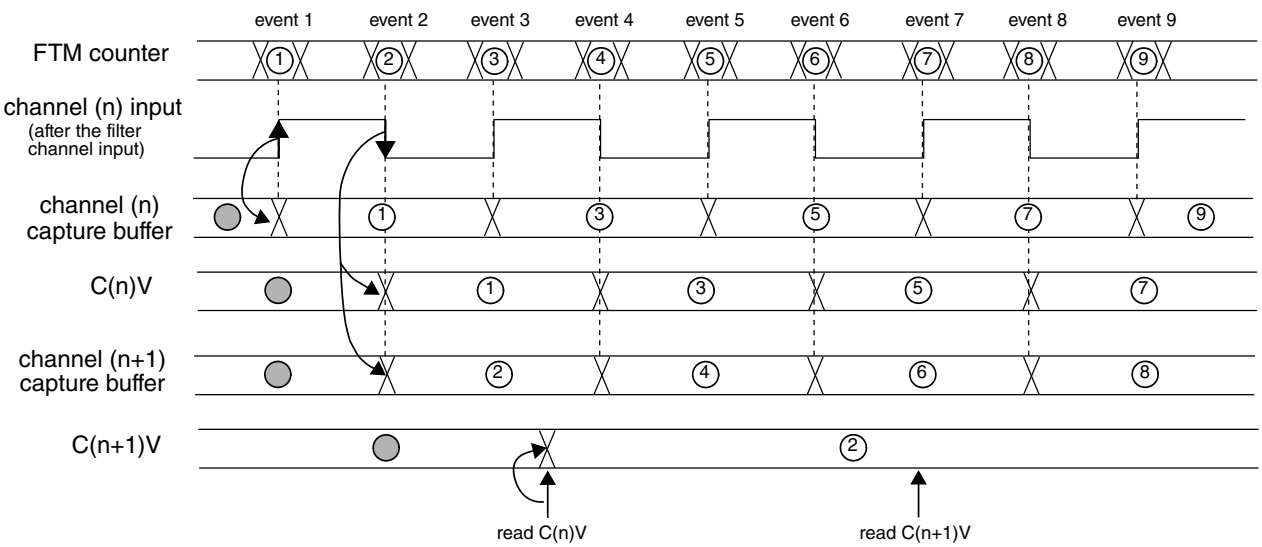


Figure 36-88. Dual Edge Capture mode read coherency mechanism

C(n)V register must be read prior to C(n+1)V register in dual edge capture one-shot and continuous modes for the read coherency mechanism works properly.

36.4.25 Quadrature Decoder mode

The Quadrature Decoder mode is selected if (QUADEN = 1). The Quadrature Decoder mode uses the input signals phase A and B to control the FTM counter increment and decrement. The following figure shows the quadrature decoder block diagram.

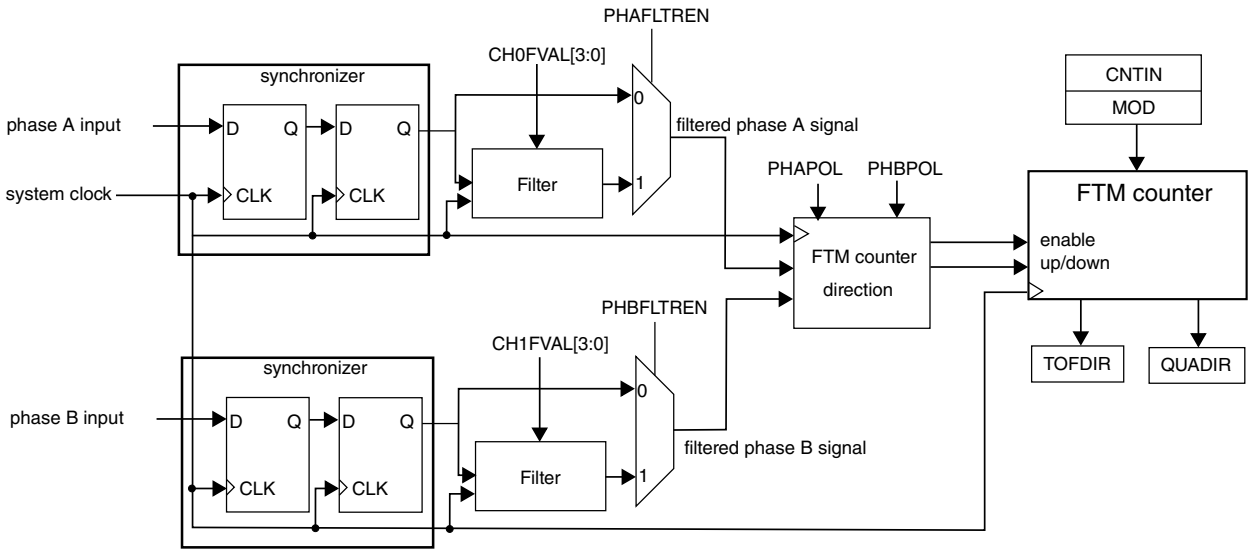


Figure 36-89. Quadrature Decoder block diagram

Each one of input signals phase A and B has a filter that is equivalent to the filter used in the channels input; Filter for Input Capture mode. The phase A input filter is enabled by PHAFLTREN bit and this filter's value is defined by CH0FVAL[3:0] bits (CH(n)FVAL[3:0] bits in FILTER0 register). The phase B input filter is enabled by PHBFLTREN bit and this filter's value is defined by CH1FVAL[3:0] bits (CH(n +1)FVAL[3:0] bits in FILTER0 register).

Except for CH0FVAL[3:0] and CH1FVAL[3:0] bits, no channel logic is used in Quadrature Decoder mode.

Note

Notice that the FTM counter is clocked by the phase A and B input signals when quadrature decoder mode is selected. Therefore it is expected that the Quadrature Decoder be used only with the FTM channels in input capture or output compare modes.

Note

An edge at phase A must not occur together an edge at phase B and vice-versa.

The PHAPOL bit selects the polarity of the phase A input, and the PHBPOL bit selects the polarity of the phase B input.

The QUADMODE selects the encoding mode used in the Quadrature Decoder mode. If QUADMODE = 1, then the count and direction encoding mode is enabled; see the following figure. In this mode, the phase B input value indicates the counting direction, and the phase A input defines the counting rate. The FTM counter is updated when there is a rising edge at phase A input signal.

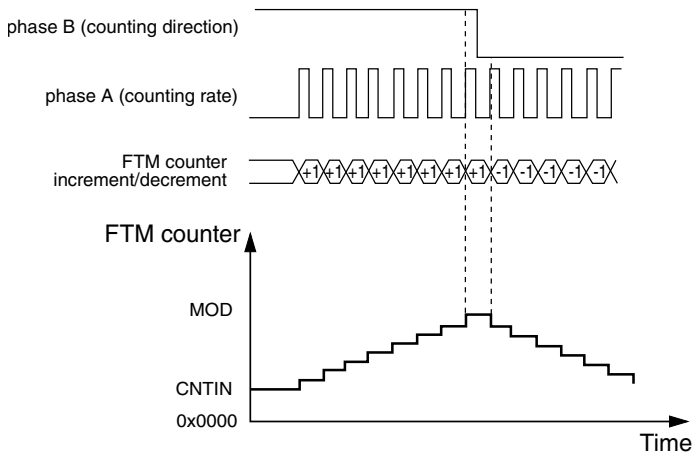


Figure 36-90. Quadrature Decoder – Count and Direction Encoding mode

If QUADMODE = 0, then the Phase A and Phase B Encoding mode is enabled; see the following figure. In this mode, the relationship between phase A and B signals indicates the counting direction, and phase A and B signals define the counting rate. The FTM counter is updated when there is an edge either at the phase A or phase B signals.

If PHAPOL = 0 and PHBPOL = 0, then the FTM counter increment happens when:

- there is a rising edge at phase A signal and phase B signal is at logic zero;
- there is a rising edge at phase B signal and phase A signal is at logic one;
- there is a falling edge at phase B signal and phase A signal is at logic zero;
- there is a falling edge at phase A signal and phase B signal is at logic one;

and the FTM counter decrement happens when:

- there is a falling edge at phase A signal and phase B signal is at logic zero;
- there is a falling edge at phase B signal and phase A signal is at logic one;
- there is a rising edge at phase B signal and phase A signal is at logic zero;
- there is a rising edge at phase A signal and phase B signal is at logic one.

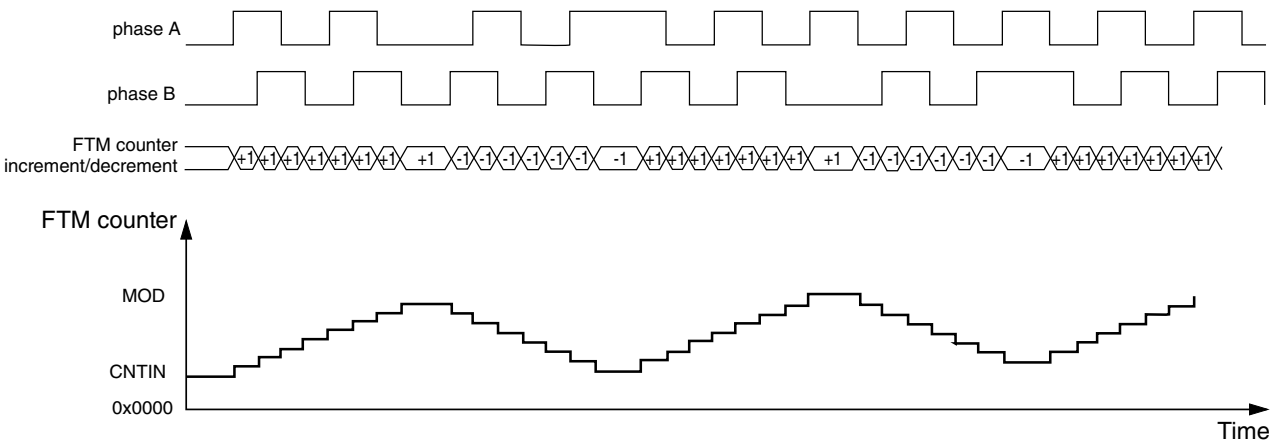


Figure 36-91. Quadrature Decoder – Phase A and Phase B Encoding mode

The following figure shows the FTM counter overflow in up counting. In this case, when the FTM counter changes from MOD to CNTIN, TOF and TOFDIR bits are set. TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was up when the FTM counter overflow occurred.

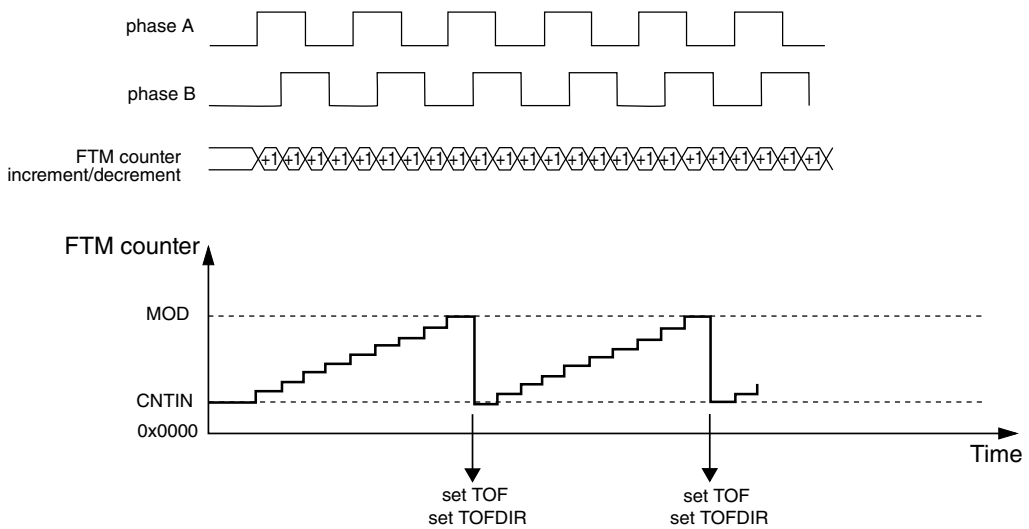


Figure 36-92. FTM Counter overflow in up counting for Quadrature Decoder mode

The following figure shows the FTM counter overflow in down counting. In this case, when the FTM counter changes from CNTIN to MOD, TOF bit is set and TOFDIR bit is cleared. TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was down when the FTM counter overflow occurred.

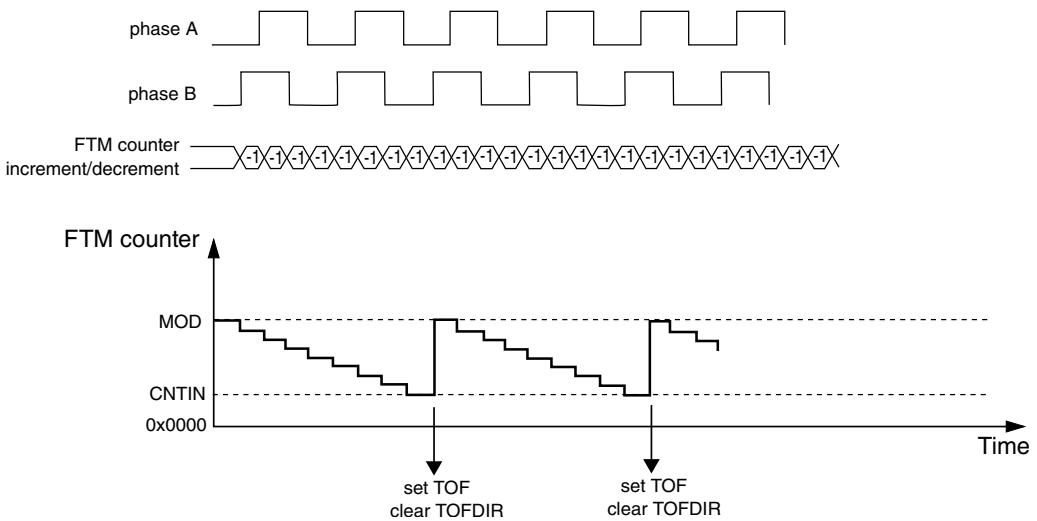


Figure 36-93. FTM counter overflow in down counting for Quadrature Decoder mode

36.4.25.1 Quadrature Decoder boundary conditions

The following figures show the FTM counter responding to motor jittering typical in motor position control applications.

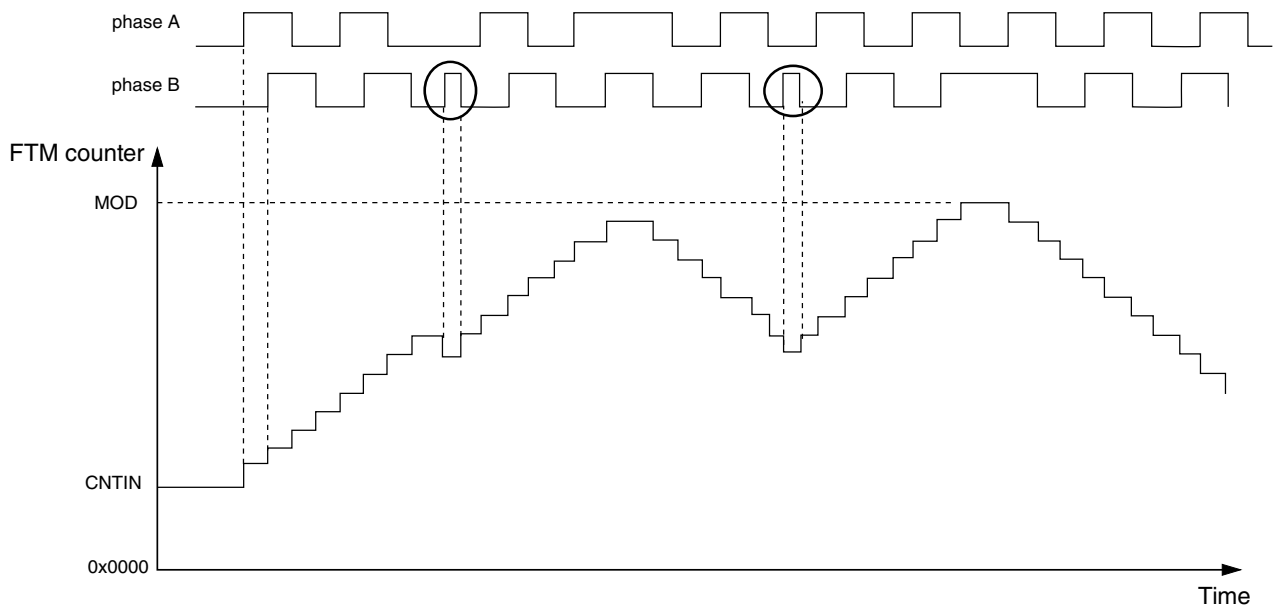


Figure 36-94. Motor position jittering in a mid count value

The following figure shows motor jittering produced by the phase B and A pulses respectively:

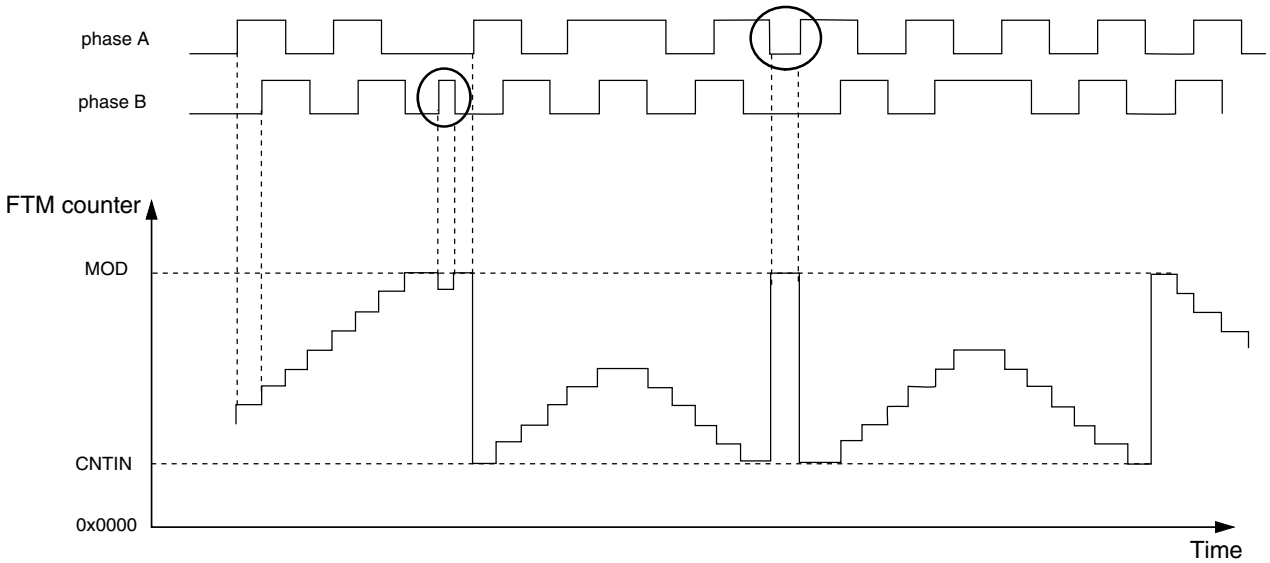


Figure 36-95. Motor position jittering near maximum and minimum count value

The first highlighted transition causes a jitter on the FTM counter value near the maximum count value (MOD). The second indicated transition occurs on phase A and causes the FTM counter transition between the maximum and minimum count values which are defined by MOD and CNTIN registers.

The appropriate settings of the phase A and phase B input filters are important to avoid glitches that may cause oscillation on the FTM counter value. The preceding figures show examples of oscillations that can be caused by poor input filter setup. Thus, it is important to guarantee a minimum pulse width to avoid these oscillations.

36.4.26 BDM mode

When the chip is in BDM mode, the BDMMODE[1:0] bits select the behavior of the FTM counter, the CH(n)F bit, the channels output, and the writes to the MOD, CNTIN, and C(n)V registers according to the following table.

Table 36-15. FTM behavior when the chip Is in BDM mode

BDMMODE	FTM Counter	CH(n)F Bit	FTM Channels Output	Writes to MOD, CNTIN, and C(n)V Registers
00	Stopped	can be set	Functional mode	Writes to these registers bypass the registers buffers
01	Stopped	is not set	The channels outputs are forced to their safe value according to POLn bit	Writes to these registers bypass the registers buffers
10	Stopped	is not set	The channels outputs are frozen when the chip enters in BDM mode	Writes to these registers bypass the registers buffers

Table continues on the next page...

Table 36-15. FTM behavior when the chip Is in BDM mode (continued)

BDMMODE	FTM Counter	CH(n)F Bit	FTM Channels Output	Writes to MOD, CNTIN, and C(n)V Registers
11	Functional mode	can be set	Functional mode	Functional mode

Note that if BDMMODE[1:0] = 2’b00 then the channels outputs remain at the value when the chip enters in BDM mode, because the FTM counter is stopped. However, the following situations modify the channels outputs in this BDM mode.

- Write any value to CNT register; see Counter reset. In this case, the FTM counter is updated with the CNTIN register value and the channels outputs are updated to the initial value – except for those channels set to Output Compare mode.
- FTM counter is reset by PWM Synchronization mode; see FTM counter synchronization. In this case, the FTM counter is updated with the CNTIN register value and the channels outputs are updated to the initial value – except for channels in Output Compare mode.
- In the channels outputs initialization, the channel (n) output is forced to the CH(n)OI bit value when the value 1 is written to INIT bit. See Initialization.

Note

The BDMMODE[1:0] = 2’b00 must not be used with the Fault control. Even if the fault control is enabled and a fault condition exists, the channels outputs values are updated as above.

Note

If CLKS[1:0] = 2'b00 in BDM, a non-zero value is written to CLKS in BDM, and CnV = CNTIN when the BDM is disabled, then the CHnF bit is set (since if the channel is a 0% EPWM signal) when the BDM is disabled.

36.4.27 Intermediate load

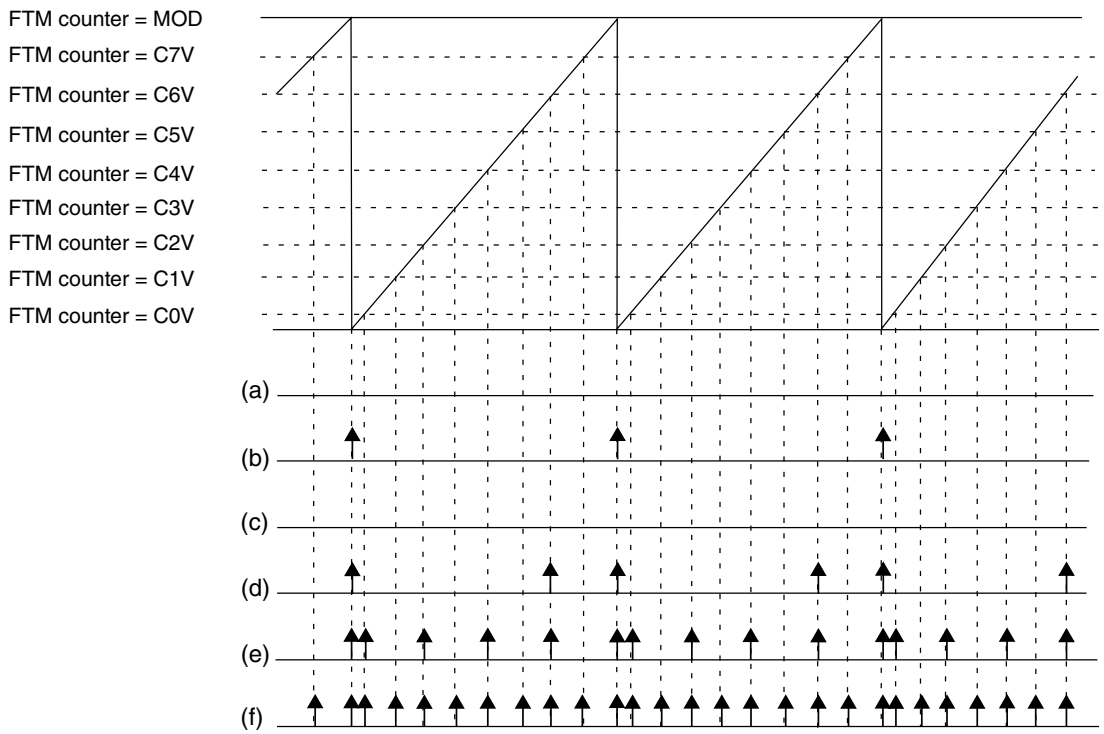
The PWMLOAD register allows to update the MOD, CNTIN, and C(n)V registers with the content of the register buffer at a defined load point. In this case, it is not required to use the PWM synchronization.

There are multiple possible loading points for intermediate load:

Table 36-16. When possible loading points are enabled

Loading point	Enabled
When the FTM counter wraps from MOD value to CNTIN value	Always
At the channel (j) match (FTM counter = C(j)V)	When CHjSEL = 1

The following figure shows some examples of enabled loading points.



NOTE

(a) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(b) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(c) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 1, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(d) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(e) LDOK = 1, CH0SEL = 1, CH1SEL = 0, CH2SEL = 1, CH3SEL = 0, CH4SEL = 1, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(f) LDOK = 1, CH0SEL = 1, CH1SEL = 1, CH2SEL = 1, CH3SEL = 1, CH4SEL = 1, CH5SEL = 1, CH6SEL = 1, CH7SEL = 1

Figure 36-96. Loading points for intermediate load

After enabling the loading points, the LDOK bit must be set for the load to occur. In this case, the load occurs at the next enabled loading point according to the following conditions:

Table 36-17. Conditions for loads occurring at the next enabled loading point

When a new value was written	Then
To the MOD register	The MOD register is updated with its write buffer value.

Table continues on the next page...

Table 36-17. Conditions for loads occurring at the next enabled loading point (continued)

When a new value was written	Then
To the CNTIN register and CNTINC = 1	The CNTIN register is updated with its write buffer value.
To the C(n)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1)	The C(n)V register is updated with its write buffer value.
To the C(n+1)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1)	The C(n+1)V register is updated with its write buffer value.

NOTE

- If ELSjB and ELSjA bits are different from zero, then the channel (j) output signal is generated according to the configured output mode. If ELSjB and ELSjA bits are zero, then the generated signal is not available on channel (j) output.
- If CHjIE = 1, then the channel (j) interrupt is generated when the channel (j) match occurs.
- At the intermediate load neither the channels outputs nor the FTM counter are changed. Software must set the intermediate load at a safe point in time.

36.4.28 Global time base (GTB)

The global time base (GTB) is a FTM function that allows the synchronization of multiple FTM modules on a chip. The following figure shows an example of the GTB feature used to synchronize two FTM modules. In this case, the FTM A and B channels can behave as if just one FTM module was used, that is, a global time base.

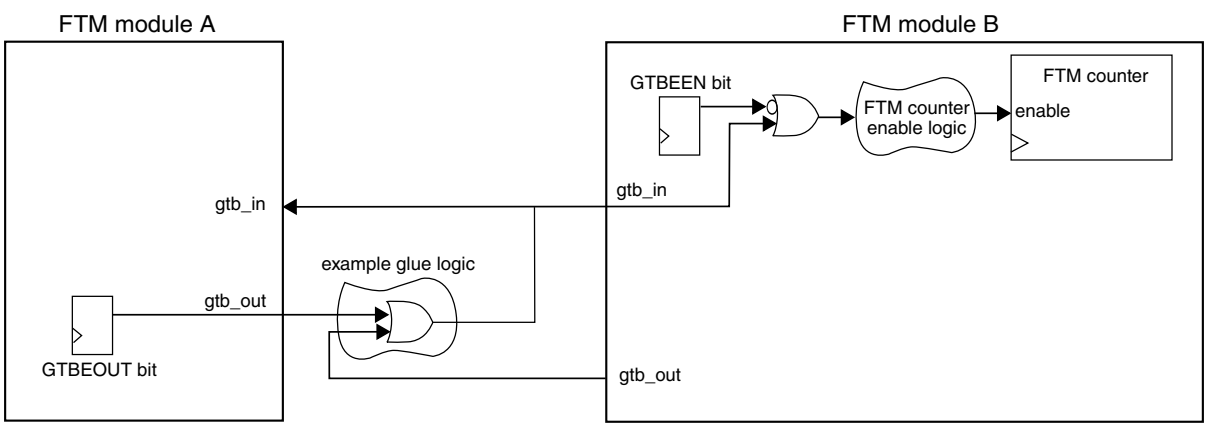


Figure 36-97. Global time base (GTB) block diagram

The GTB functionality is implemented by the GTBEEN and GTBEOUT bits in the CONF register, the input signal *gtb_in*, and the output signal *gtb_out*. The GTBEEN bit enables gtb_in to control the FTM counter enable signal:

- If GTBEEN = 0, each one of FTM modules works independently according to their configured mode.
- If GTBEEN = 1, the FTM counter update is enabled only when gtb_in is 1.

In the configuration described in the preceding figure, FTM modules A and B have their FTM counters enabled if at least one of the gtb_out signals from one of the FTM modules is 1. There are several possible configurations for the interconnection of the gtb_in and gtb_out signals, represented by the example glue logic shown in the figure. Note that these configurations are chip-dependent and implemented outside of the FTM modules. See the chip-specific FTM information for the chip's specific implementation.

NOTE

- In order to use the GTB signals to synchronize the FTM counter of different FTM modules, the configuration of each FTM module should guarantee that its FTM counter starts counting as soon as the gtb_in signal is 1.
- The GTB feature does not provide continuous synchronization of FTM counters, meaning that the FTM counters may lose synchronization during FTM operation. The GTB feature only allows the FTM counters to *start* their operation synchronously.

36.4.28.1 Enabling the global time base (GTB)

To enable the GTB feature, follow these steps for each participating FTM module:

1. Stop the FTM counter: Write 00b to SC[CLKS].
2. Program the FTM to the intended configuration. The FTM counter mode needs to be consistent across all participating modules.
3. Write 1 to CONF[GTBEEN] and write 0 to CONF[GTBEOUT] at the same time.
4. Select the intended FTM counter clock source in SC[CLKS]. The clock source needs to be consistent across all participating modules.
5. Reset the FTM counter: Write any value to the CNT register.

To initiate the GTB feature in the configuration described in the preceding figure, write 1 to CONF[GTBEOUT] in the FTM module used as the time base.

36.5 Reset overview

The FTM is reset whenever any chip reset occurs.

When the FTM exits from reset:

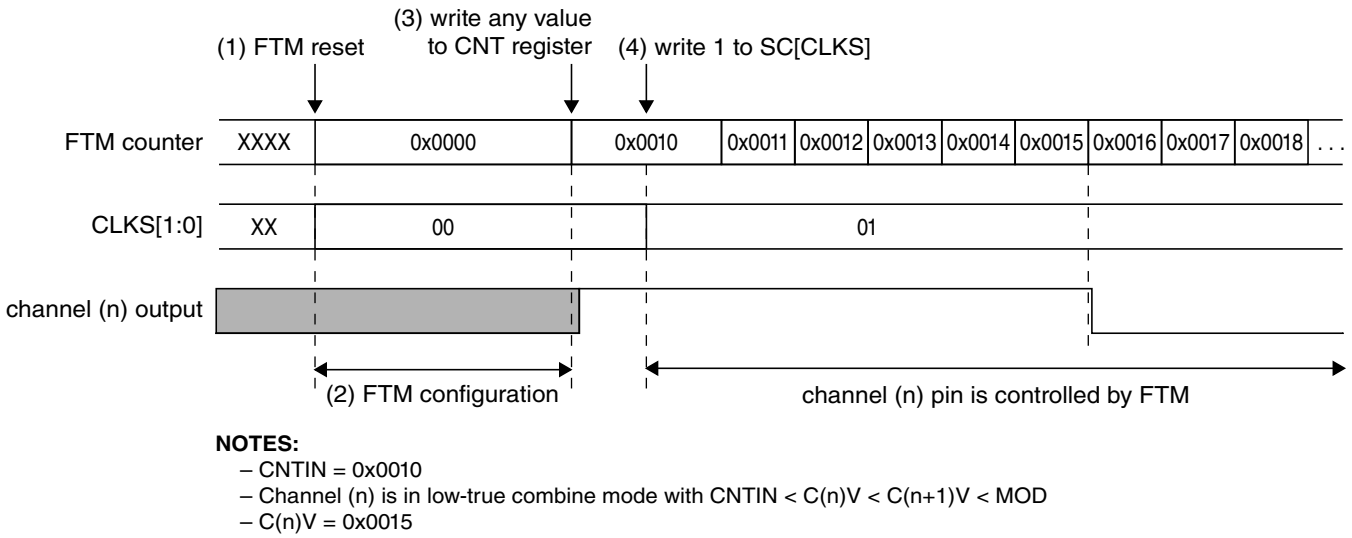
- the FTM counter and the prescaler counter are zero and are stopped (CLKS[1:0] = 00b);
- the timer overflow interrupt is zero, see Timer Overflow Interrupt;
- the channels interrupts are zero, see Channel (n) Interrupt;
- the fault interrupt is zero, see Fault Interrupt;
- the channels are in input capture mode, see Input Capture mode;
- the channels outputs are zero;
- the channels pins are not controlled by FTM (ELS(n)B:ELS(n)A = 0:0) (See the table in the description of CnSC register).

The following figure shows the FTM behavior after the reset. At the reset (item 1), the FTM counter is disabled (see the description of the CLKS field in the Status and Control register), its value is updated to zero and the pins are not controlled by FTM (See the table in the description of CnSC register).

After the reset, the FTM should be configurated (item 2). It is necessary to define the FTM counter mode, the FTM counting limits (MOD and CNTIN registers value), the channels mode and CnV registers value according to the channels mode.

Thus, it is recommended to write any value to CNT register (item 3). This write updates the FTM counter with the CNTIN register value and the channels output with its initial value (except for channels in output compare mode) (Counter reset).

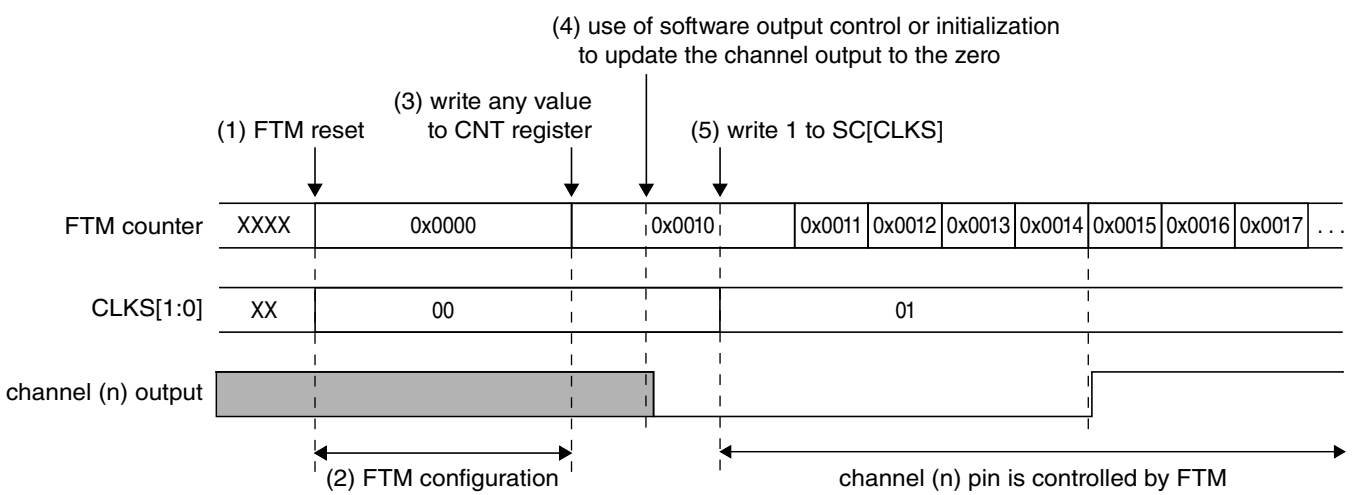
The next step is to select the FTM counter clock by the CLKS[1:0] bits (item 4). It is important to highlight that the pins are only controlled by FTM when CLKS[1:0] bits are different from zero (See the table in the description of CnSC register).



NOTES:
– CNTIN = 0x0010
– Channel (n) is in low-true combine mode with CNTIN < C(n)V < C(n+1)V < MOD
– C(n)V = 0x0015

Figure 36-98. FTM behavior after reset when the channel (n) is in Combine mode

The following figure shows an example when the channel (n) is in Output Compare mode and the channel (n) output is toggled when there is a match. In the Output Compare mode, the channel output is not updated to its initial value when there is a write to CNT register (item 3). In this case, use the software output control (Software output control) or the initialization (Initialization) to update the channel output to the selected value (item 4).



NOTES:
– CNTIN = 0x0010
– Channel (n) is in output compare and the channel (n) output is toggled when there is a match
– C(n)V = 0x0014

Figure 36-99. FTM behavior after reset when the channel (n) is in Output Compare mode

36.6 FTM Interrupts

36.6.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

36.6.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

36.6.3 Fault Interrupt

The fault interrupt is generated when (FAULTIE = 1) and (FAULTF = 1).

36.7 Initialization Procedure

The following initialization procedure is recommended to configure the FlexTimer operation. This procedure can also be used to do a new configuration of the FlexTimer operation.

- Define the POL bits.
- Mask the channels outputs using SYNCHOM = 0. Two clocks after the write to OUTMASK, the channels output are in the safe value.
- (Re)Configuration FTM counter and channels to generation of periodic signals - Disable the clock. If the selected mode is Quadrature Decoder, then disable this mode. Examples of the (re)configuration:
  - Write to MOD.
  - Write to CNTIN.
  - Select OC, EPWM, CPWM, Combine, Complement modes for all channels that will be used
  - Select the high-true and low-true channels modes.
  - Write to CnV for all channels that will be used .
  - (Re)Configure deadtime and fault control.
  - Do not use the SWOC without SW synchronization (see item 6).
  - Do not use the Inverting without SW synchronization (see item 6).
  - Do not use the Initialization.
  - Do not change the polarity control.
  - Do not configure the HW synchronization

- Write any value to CNT. The FTM Counter is reset and the channels output are updated according to new configuration.
- Enable the clock. Write to CLKS[1:0] bits a value different from zero. If in the Quadrature Decoder mode, enable this mode.
- Configure the SW synchronization for SWOC (if it is necessary), Inverting (if it is necessary) and Output Mask (always)
  - Select synchronization for Output Mask Write to SYNC (SWSYNC = 0, TRIG2 = 0, TRIG1 = 0, TRIG0 = 0, SYNCHOM = 1, REINIT = 0, CNTMAX = 0, CNTMIN = 0)
  - Write to SYNCONF.
    - HW Synchronization can not be enabled (HWSOC = 0, HWINVC = 0, HWOM = 0, HWWRBUF = 0, HWRSTCNT = 0, HWTRIGMODE = 0).
    - SW Synchronization for SWOC (if it is necessary): SWSOC = [0/1] and SWOC = [0/1].
    - SW Synchronization for Inverting (if it is necessary): SWINVC = [0/1] and INVC = [0/1].
    - SW Synchronization for SWOM (always): SWOM = 1. No enable the SW Synchronization for write buffers (because the writes to registers with write buffer are done using CLKS[1:0] = 2'b00): SWWRBUF = 0 and CNTINC = 0 .
    - SW Synchronization for counter reset (always): SWRSTCNT = 1.
    - Enhanced synchronization (always): SYNCMODE = 1
  - If the SWOC is used (SWSOC = 1 and SWOC = 1), then write to SWOCTRL register.
  - If the Inverting is used (SWINVC = 1 and INVC = 1), then write to INVCTRL register.
  - Write to OUTMASK to enable the masked channels.
- Generate the Software Trigger Write to SYNC (SWSYNC = 1, TRIG2 = 0, TRIG1 = 0, TRIG0 = 0, SYNCHOM = 1, REINIT = 0, CNTMAX = 0, CNTMIN = 0)

Chapter 37
Low-Power Timer (LPTMR)

37.1 Introduction

NOTE

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

The low-power timer (LPTMR) can be configured to operate as a time counter with optional prescaler, or as a pulse counter with optional glitch filter, across all power modes, including the low-leakage modes. It can also continue operating through most system reset events, allowing it to be used as a time of day counter.

37.1.1 Features

The features of the LPTMR module include:

- 16-bit time counter or pulse counter with compare
  - Optional interrupt can generate asynchronous wakeup from any low-power mode
  - Hardware trigger output
  - Counter supports free-running mode or reset on compare
- Configurable clock source for prescaler/glitch filter
- Configurable input source for pulse counter
  - Rising-edge or falling-edge

37.1.2 Modes of operation

The following table describes the operation of the LPTMR module in various modes.

Table 37-1. Modes of operation

Modes	Description
Run	The LPTMR operates normally.
Wait	The LPTMR continues to operate normally and may be configured to exit the low-power mode by generating an interrupt request.
Stop	The LPTMR continues to operate normally and may be configured to exit the low-power mode by generating an interrupt request.
Low-Leakage	The LPTMR continues to operate normally and may be configured to exit the low-power mode by generating an interrupt request.
Debug	The LPTMR operates normally in Pulse Counter mode, but counter does not increment in Time Counter mode.

37.2 LPTMR signal descriptions

Table 37-2. LPTMR signal descriptions

Signal	I/O	Description
LPTMR0_ALT*n*	I	Pulse Counter Input pin

37.2.1 Detailed signal descriptions

Table 37-3. LPTMR interface—detailed signal descriptions

Signal	I/O	Description	
LPTMR_ALT*n*	I	Pulse Counter Input The LPTMR can select one of the input pins to be used in Pulse Counter mode.	
		State meaning	Assertion—If configured for pulse counter mode with active-high input, then assertion causes the CNR to increment. Deassertion—If configured for pulse counter mode with active-low input, then deassertion causes the CNR to increment.
		Timing	Assertion or deassertion may occur at any time; input may assert asynchronously to the bus clock.

37.3 Memory map and register definition

LPTMR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_0000	Low Power Timer Control Status Register (LPTMR0_CSR)	32	R/W	0000_0000h	37.3.1/875
4004_0004	Low Power Timer Prescale Register (LPTMR0_PSR)	32	R/W	0000_0000h	37.3.2/876
4004_0008	Low Power Timer Compare Register (LPTMR0_CMR)	32	R/W	0000_0000h	37.3.3/878
4004_000C	Low Power Timer Counter Register (LPTMR0_CNR)	32	R/W	0000_0000h	37.3.4/878

37.3.1 Low Power Timer Control Status Register (LPTMRx_CSR)

Address: 4004_0000h base + 0h offset = 4004_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								TCF	TIE	TPS		TPP	TFC	TMS	TEN
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LPTMRx_CSR field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TCF	Timer Compare Flag TCF is set when the LPTMR is enabled and the CNR equals the CMR and increments. TCF is cleared when the LPTMR is disabled or a logic 1 is written to it. 0 The value of CNR is not equal to CMR and increments. 1 The value of CNR is equal to CMR and increments.
6 TIE	Timer Interrupt Enable When TIE is set, the LPTMR Interrupt is generated whenever TCF is also set. 0 Timer interrupt disabled. 1 Timer interrupt enabled.
5–4 TPS	Timer Pin Select Configures the input source to be used in Pulse Counter mode. TPS must be altered only when the LPTMR is disabled. The input connections vary by device. See the chip configuration details for information on the connections to these inputs. 00 Pulse counter input 0 is selected.

Table continues on the next page...

LPTMRx_CSR field descriptions (continued)

Field	Description
	01 Pulse counter input 1 is selected. 10 Pulse counter input 2 is selected. 11 Pulse counter input 3 is selected.
3 TPP	Timer Pin Polarity Configures the polarity of the input source in Pulse Counter mode. TPP must be changed only when the LPTMR is disabled. 0 Pulse Counter input source is active-high, and the CNR will increment on the rising-edge. 1 Pulse Counter input source is active-low, and the CNR will increment on the falling-edge.
2 TFC	Timer Free-Running Counter When clear, TFC configures the CNR to reset whenever TCF is set. When set, TFC configures the CNR to reset on overflow. TFC must be altered only when the LPTMR is disabled. 0 CNR is reset whenever TCF is set. 1 CNR is reset on overflow.
1 TMS	Timer Mode Select Configures the mode of the LPTMR. TMS must be altered only when the LPTMR is disabled. 0 Time Counter mode. 1 Pulse Counter mode.
0 TEN	Timer Enable When TEN is clear, it resets the LPTMR internal logic, including the CNR and TCF. When TEN is set, the LPTMR is enabled. While writing 1 to this field, CSR[5:1] must not be altered. 0 LPTMR is disabled and internal logic is reset. 1 LPTMR is enabled.

37.3.2 Low Power Timer Prescale Register (LPTMRx_PSR)

Address: 4004_0000h base + 4h offset = 4004_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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								PRESCALE					PBYP	PCS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LPTMRx_PSR field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–3 PRESCALE	Prescale Value Configures the size of the Prescaler in Time Counter mode or width of the glitch filter in Pulse Counter mode. PRESCALE must be altered only when the LPTMR is disabled. 0000 Prescaler divides the prescaler clock by 2; glitch filter does not support this configuration. 0001 Prescaler divides the prescaler clock by 4; glitch filter recognizes change on input pin after 2 rising clock edges. 0010 Prescaler divides the prescaler clock by 8; glitch filter recognizes change on input pin after 4 rising clock edges. 0011 Prescaler divides the prescaler clock by 16; glitch filter recognizes change on input pin after 8 rising clock edges. 0100 Prescaler divides the prescaler clock by 32; glitch filter recognizes change on input pin after 16 rising clock edges. 0101 Prescaler divides the prescaler clock by 64; glitch filter recognizes change on input pin after 32 rising clock edges. 0110 Prescaler divides the prescaler clock by 128; glitch filter recognizes change on input pin after 64 rising clock edges. 0111 Prescaler divides the prescaler clock by 256; glitch filter recognizes change on input pin after 128 rising clock edges. 1000 Prescaler divides the prescaler clock by 512; glitch filter recognizes change on input pin after 256 rising clock edges. 1001 Prescaler divides the prescaler clock by 1024; glitch filter recognizes change on input pin after 512 rising clock edges. 1010 Prescaler divides the prescaler clock by 2048; glitch filter recognizes change on input pin after 1024 rising clock edges. 1011 Prescaler divides the prescaler clock by 4096; glitch filter recognizes change on input pin after 2048 rising clock edges. 1100 Prescaler divides the prescaler clock by 8192; glitch filter recognizes change on input pin after 4096 rising clock edges. 1101 Prescaler divides the prescaler clock by 16,384; glitch filter recognizes change on input pin after 8192 rising clock edges. 1110 Prescaler divides the prescaler clock by 32,768; glitch filter recognizes change on input pin after 16,384 rising clock edges. 1111 Prescaler divides the prescaler clock by 65,536; glitch filter recognizes change on input pin after 32,768 rising clock edges.
2 PBYP	Prescaler Bypass When PBYP is set, the selected prescaler clock in Time Counter mode or selected input source in Pulse Counter mode directly clocks the CNR. When PBYP is clear, the CNR is clocked by the output of the prescaler/glitch filter. PBYP must be altered only when the LPTMR is disabled. 0 Prescaler/glitch filter is enabled. 1 Prescaler/glitch filter is bypassed.
PCS	Prescaler Clock Select Selects the clock to be used by the LPTMR prescaler/glitch filter. PCS must be altered only when the LPTMR is disabled. The clock connections vary by device. **NOTE:** See the chip configuration details for information on the connections to these inputs.

Table continues on the next page...

LPTMRx_PSR field descriptions (continued)

Field	Description
	00 Prescaler/glitch filter clock 0 selected. 01 Prescaler/glitch filter clock 1 selected. 10 Prescaler/glitch filter clock 2 selected. 11 Prescaler/glitch filter clock 3 selected.

37.3.3 Low Power Timer Compare Register (LPTMRx_CMR)

Address: 4004_0000h base + 8h offset = 4004_0008h

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																COMPARE															
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	0	0	0

LPTMRx_CMR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COMPARE	Compare Value When the LPTMR is enabled and the CNR equals the value in the CMR and increments, TCF is set and the hardware trigger asserts until the next time the CNR increments. If the CMR is 0, the hardware trigger will remain asserted until the LPTMR is disabled. If the LPTMR is enabled, the CMR must be altered only when TCF is set.

37.3.4 Low Power Timer Counter Register (LPTMRx_CNR)

NOTE

See LPTMR counter for details on how to read counter value.

Address: 4004_0000h base + Ch offset = 4004_000Ch

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																COUNTER															
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	0	0	0

LPTMRx_CNR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LPTMRx_CNR field descriptions (continued)

Field	Description
COUNTER	Counter Value

37.4 Functional description

37.4.1 LPTMR power and reset

The LPTMR remains powered in all power modes, including low-leakage modes. If the LPTMR is not required to remain operating during a low-power mode, then it must be disabled before entering the mode.

The LPTMR is reset only on global Power On Reset (POR) or Low Voltage Detect (LVD). When configuring the LPTMR registers, the CSR must be initially written with the timer disabled, before configuring the PSR and CMR. Then, CSR[TIE] must be set as the last step in the initialization. This ensures the LPTMR is configured correctly and the LPTMR counter is reset to zero following a warm reset.

37.4.2 LPTMR clocking

The LPTMR prescaler/glitch filter can be clocked by one of the four clocks. The clock source must be enabled before the LPTMR is enabled.

NOTE

The clock source selected need to be configured to remain enabled in low-power modes, otherwise the LPTMR will not operate during low-power modes.

In Pulse Counter mode with the prescaler/glitch filter bypassed, the selected input source directly clocks the CNR and no other clock source is required. To minimize power in this case, configure the prescaler clock source for a clock that is not toggling.

NOTE

The clock source or pulse input source selected for the LPTMR should not exceed the frequency f_{LPTMR} defined in the device datasheet.

37.4.3 LPTMR prescaler/glitch filter

The LPTMR prescaler and glitch filter share the same logic which operates as a prescaler in Time Counter mode and as a glitch filter in Pulse Counter mode.

NOTE

The prescaler/glitch filter configuration must not be altered when the LPTMR is enabled.

37.4.3.1 Prescaler enabled

In Time Counter mode, when the prescaler is enabled, the output of the prescaler directly clocks the CNR. When the LPTMR is enabled, the CNR will increment every 2^2 to 2^{16} prescaler clock cycles. After the LPTMR is enabled, the first increment of the CNR will take an additional one or two prescaler clock cycles due to synchronization logic.

37.4.3.2 Prescaler bypassed

In Time Counter mode, when the prescaler is bypassed, the selected prescaler clock increments the CNR on every clock cycle. When the LPTMR is enabled, the first increment will take an additional one or two prescaler clock cycles due to synchronization logic.

37.4.3.3 Glitch filter

In Pulse Counter mode, when the glitch filter is enabled, the output of the glitch filter directly clocks the CNR. When the LPTMR is first enabled, the output of the glitch filter is asserted, that is, logic 1 for active-high and logic 0 for active-low. The following table shows the change in glitch filter output with the selected input source.

If	Then
The selected input source remains deasserted for at least 2^1 to 2^{15} consecutive prescaler clock rising edges	The glitch filter output will also deassert.
The selected input source remains asserted for at least 2^1 to 2^{15} consecutive prescaler clock rising-edges	The glitch filter output will also assert.

NOTE

The input is only sampled on the rising clock edge.

The CNR will increment each time the glitch filter output asserts. In Pulse Counter mode, the maximum rate at which the CNR can increment is once every 2^2 to 2^{16} prescaler clock edges. When first enabled, the glitch filter will wait an additional one or two prescaler clock edges due to synchronization logic.

37.4.3.4 Glitch filter bypassed

In Pulse Counter mode, when the glitch filter is bypassed, the selected input source increments the CNR every time it asserts. Before the LPTMR is first enabled, the selected input source is forced to be asserted. This prevents the CNR from incrementing if the selected input source is already asserted when the LPTMR is first enabled.

37.4.4 LPTMR compare

When the CNR equals the value of the CMR and increments, the following events occur:

- CSR[TCF] is set.
- LPTMR interrupt is generated if CSR[TIE] is also set.
- LPTMR hardware trigger is generated.
- CNR is reset if CSR[TFC] is clear.

When the LPTMR is enabled, the CMR can be altered only when CSR[TCF] is set. When updating the CMR, the CMR must be written and CSR[TCF] must be cleared before the LPTMR counter has incremented past the new LPTMR compare value.

37.4.5 LPTMR counter

The CNR increments by one on every:

- Prescaler clock in Time Counter mode with prescaler bypassed
- Prescaler output in Time Counter mode with prescaler enabled
- Input source assertion in Pulse Counter mode with glitch filter bypassed
- Glitch filter output in Pulse Counter mode with glitch filter enabled

The CNR is reset when the LPTMR is disabled or if the counter register overflows. If CSR[TFC] is cleared, then the CNR is also reset whenever CSR[TCF] is set.

The CNR continues incrementing when the core is halted in Debug mode when configured for Pulse Counter mode, the CNR will stop incrementing when the core is halted in Debug mode when configured for Time Counter mode.

The CNR cannot be initialized, but can be read at any time. On each read of the CNR, software must first write to the CNR with any value. This will synchronize and register the current value of the CNR into a temporary register. The contents of the temporary register are returned on each read of the CNR.

When reading the CNR, the bus clock must be at least two times faster than the rate at which the LPTMR counter is incrementing, otherwise incorrect data may be returned.

37.4.6 LPTMR hardware trigger

The LPTMR hardware trigger asserts at the same time the CSR[TCF] is set and can be used to trigger hardware events in other peripherals without software intervention. The hardware trigger is always enabled.

When	Then
The CMR is set to 0 with CSR[TFC] clear	The LPTMR hardware trigger will assert on the first compare and does not deassert.
The CMR is set to a nonzero value, or, if CSR[TFC] is set	The LPTMR hardware trigger will assert on each compare and deassert on the following increment of the CNR.

37.4.7 LPTMR interrupt

The LPTMR interrupt is generated whenever CSR[TIE] and CSR[TCF] are set. CSR[TCF] is cleared by disabling the LPTMR or by writing a logic 1 to it.

CSR[TIE] can be altered and CSR[TCF] can be cleared while the LPTMR is enabled.

The LPTMR interrupt is generated asynchronously to the system clock and can be used to generate a wakeup from any low-power mode, including the low-leakage modes, provided the LPTMR is enabled as a wakeup source.

Chapter 38
Programmable Delay Block (PDB)

38.1 Introduction

NOTE

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

The Programmable Delay Block (PDB) provides controllable delays from either an internal or an external trigger, or a programmable interval tick, to the hardware trigger inputs of ADCs and/or generates the interval triggers to DACs, so that the precise timing between ADC conversions and/or DAC updates can be achieved. The PDB can optionally provide pulse outputs (Pulse-Out's) that are used as the sample window in the CMP block.

38.1.1 Features

- Up to 15 trigger input sources and one software trigger source
- Up to 8 configurable PDB channels for ADC hardware trigger
  - One PDB channel is associated with one ADC
  - One trigger output for ADC hardware trigger and up to 8 pre-trigger outputs for ADC trigger select per PDB channel
  - Trigger outputs can be enabled or disabled independently
  - One 16-bit delay register per pre-trigger output
  - Optional bypass of the delay registers of the pre-trigger outputs
  - Operation in One-Shot or Continuous modes

    - Optional back-to-back mode operation, which enables the ADC conversions complete to trigger the next PDB channel
    - One programmable delay interrupt
    - One sequence error interrupt
    - One channel flag and one sequence error flag per pre-trigger
    - DMA support
- Up to 8 DAC interval triggers
    - One interval trigger output per DAC
    - One 16-bit delay interval register per DAC trigger output
    - Optional bypass of the delay interval trigger registers
    - Optional external triggers
- Up to 8 pulse outputs (pulse-out's)
    - Pulse-out's can be enabled or disabled independently
    - Programmable pulse width

NOTE

The number of PDB input and output triggers are chip-specific. See the chip-specific PDB information for details.

38.1.2 Implementation

In this section, the following letters refer to the number of output triggers:

- N—Total available number of PDB channels.
- n—PDB channel number, valid from 0 to *N*-1.
- M—Total available pre-trigger per PDB channel.
- m—Pre-trigger number, valid from 0 to *M*-1.
- X—Total number of DAC interval triggers.
- x—DAC interval trigger output number, valid from 0 to *X*-1.

- Y—Total number of Pulse-Out's.
- y—Pulse-Out number, valid value is from 0 to *Y*-1.

NOTE

The number of module output triggers to core is chip-specific. For module to core output triggers implementation, see the chip configuration information.

38.1.3 Back-to-back acknowledgment connections

PDB back-to-back operation acknowledgment connections are chip-specific. For implementation, see the chip configuration information.

38.1.4 DAC External Trigger Input Connections

The implementation of DAC external trigger inputs is chip-specific. See the chip configuration information for details.

38.1.5 Block diagram

This diagram illustrates the major components of the PDB.

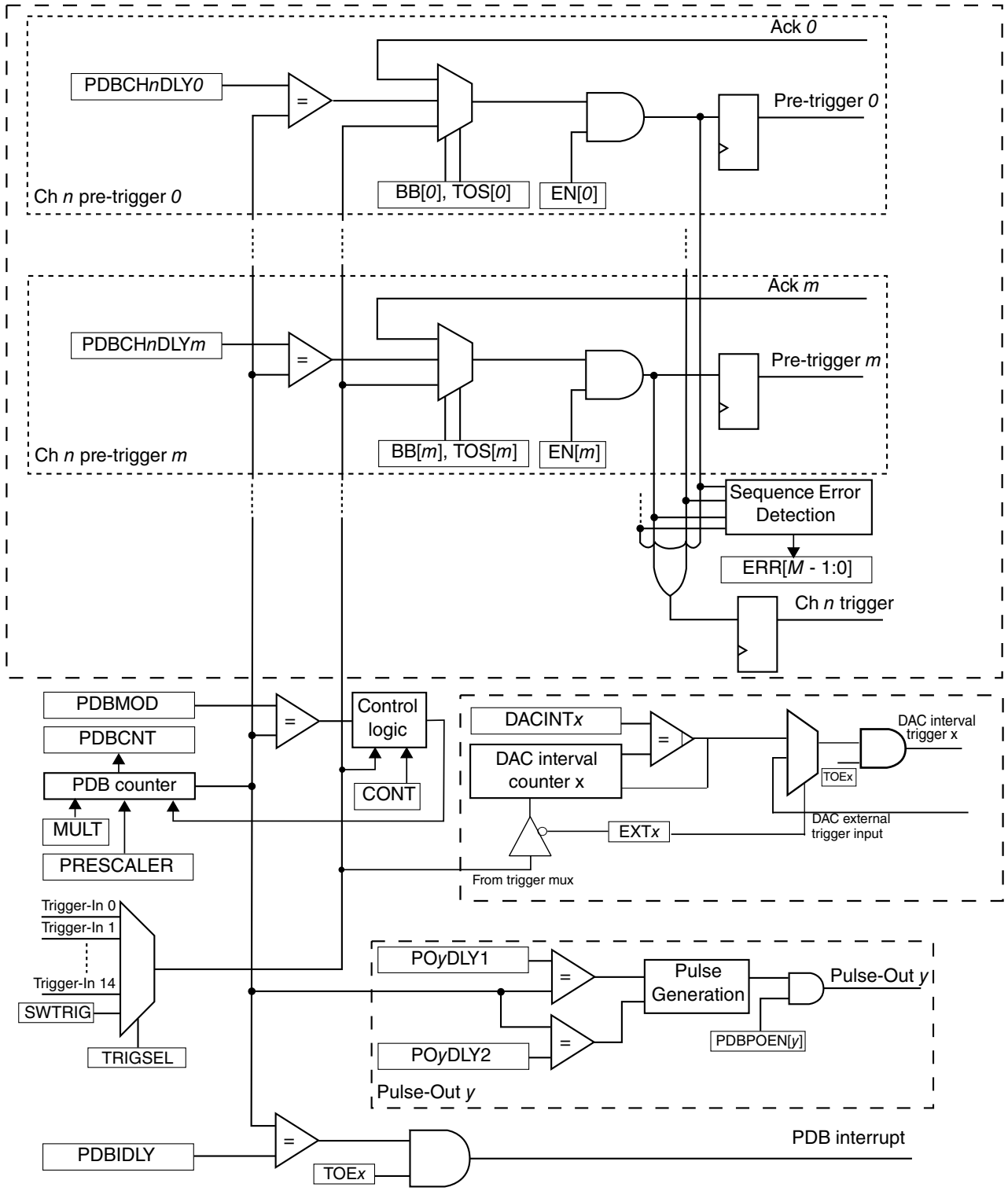


Figure 38-1. PDB block diagram

In this diagram, only one PDB channel *n*, one DAC interval trigger *x*, and one Pulse-Out *y* are shown. The PDB-enabled control logic and the sequence error interrupt logic are not shown.

38.1.6 Modes of operation

PDB ADC trigger operates in the following modes:

- Disabled—Counter is off, all pre-trigger and trigger outputs are low if PDB is not in back-to-back operation of Bypass mode.
- Debug—Counter is paused when processor is in Debug mode, and the counter for the DAC trigger is also paused in Debug mode.
- Enabled One-Shot—Counter is enabled and restarted at count zero upon receiving a positive edge on the selected trigger input source or software trigger is selected and SC[SWTRIG] is written with 1. In each PDB channel, an enabled pre-trigger asserts once per trigger input event. The trigger output asserts whenever any of the pre-triggers is asserted.
- Enabled Continuous—Counter is enabled and restarted at count zero. The counter is rolled over to zero again when the count reaches the value specified in the modulus register, and the counting is restarted. This enables a continuous stream of pre-triggers/trigger outputs as a result of a single trigger input event.
- Enabled Bypassed—The pre-trigger and trigger outputs assert immediately after a positive edge on the selected trigger input source or software trigger is selected and SC[SWTRIG] is written with 1, that is the delay registers are bypassed. It is possible to bypass any one or more of the delay registers; therefore, this mode can be used in conjunction with One-Shot or Continuous mode.

38.2 PDB signal descriptions

This table shows the detailed description of the external signal.

Table 38-1. PDB signal descriptions

Signal	Description	I/O
EXTRG	External Trigger Input Source If the PDB is enabled and external trigger input source is selected, a positive edge on the EXTRG signal resets and starts the counter.	I

38.3 Memory map and register definition

PDB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_1000	Status and Control register (PDB1_SC)	32	R/W	0000_0000h	38.3.1/889
4003_1004	Modulus register (PDB1_MOD)	32	R/W	0000_FFFFh	38.3.2/892
4003_1008	Counter register (PDB1_CNT)	32	R	0000_0000h	38.3.3/892
4003_100C	Interrupt Delay register (PDB1_IDLY)	32	R/W	0000_FFFFh	38.3.4/893
4003_1010	Channel n Control register 1 (PDB1_CH0C1)	32	R/W	0000_0000h	38.3.5/893
4003_1014	Channel n Status register (PDB1_CH0S)	32	R/W	0000_0000h	38.3.6/894
4003_1018	Channel n Delay 0 register (PDB1_CH0DLY0)	32	R/W	0000_0000h	38.3.7/895
4003_101C	Channel n Delay 1 register (PDB1_CH0DLY1)	32	R/W	0000_0000h	38.3.8/896
4003_1038	Channel n Control register 1 (PDB1_CH1C1)	32	R/W	0000_0000h	38.3.5/893
4003_103C	Channel n Status register (PDB1_CH1S)	32	R/W	0000_0000h	38.3.6/894
4003_1040	Channel n Delay 0 register (PDB1_CH1DLY0)	32	R/W	0000_0000h	38.3.7/895
4003_1044	Channel n Delay 1 register (PDB1_CH1DLY1)	32	R/W	0000_0000h	38.3.8/896
4003_1150	DAC Interval Trigger n Control register (PDB1_DACINTC0)	32	R/W	0000_0000h	38.3.9/896
4003_1154	DAC Interval n register (PDB1_DACINT0)	32	R/W	0000_0000h	38.3.10/ 897
4003_1190	Pulse-Out n Enable register (PDB1_POEN)	32	R/W	0000_0000h	38.3.11/ 898
4003_1194	Pulse-Out n Delay register (PDB1_PO0DLY)	32	R/W	0000_0000h	38.3.12/ 898
4003_1198	Pulse-Out n Delay register (PDB1_PO1DLY)	32	R/W	0000_0000h	38.3.12/ 898
4003_6000	Status and Control register (PDB0_SC)	32	R/W	0000_0000h	38.3.1/889
4003_6004	Modulus register (PDB0_MOD)	32	R/W	0000_FFFFh	38.3.2/892
4003_6008	Counter register (PDB0_CNT)	32	R	0000_0000h	38.3.3/892
4003_600C	Interrupt Delay register (PDB0_IDLY)	32	R/W	0000_FFFFh	38.3.4/893
4003_6010	Channel n Control register 1 (PDB0_CH0C1)	32	R/W	0000_0000h	38.3.5/893
4003_6014	Channel n Status register (PDB0_CH0S)	32	R/W	0000_0000h	38.3.6/894
4003_6018	Channel n Delay 0 register (PDB0_CH0DLY0)	32	R/W	0000_0000h	38.3.7/895
4003_601C	Channel n Delay 1 register (PDB0_CH0DLY1)	32	R/W	0000_0000h	38.3.8/896
4003_6038	Channel n Control register 1 (PDB0_CH1C1)	32	R/W	0000_0000h	38.3.5/893
4003_603C	Channel n Status register (PDB0_CH1S)	32	R/W	0000_0000h	38.3.6/894
4003_6040	Channel n Delay 0 register (PDB0_CH1DLY0)	32	R/W	0000_0000h	38.3.7/895
4003_6044	Channel n Delay 1 register (PDB0_CH1DLY1)	32	R/W	0000_0000h	38.3.8/896
4003_6150	DAC Interval Trigger n Control register (PDB0_DACINTC0)	32	R/W	0000_0000h	38.3.9/896
4003_6154	DAC Interval n register (PDB0_DACINT0)	32	R/W	0000_0000h	38.3.10/ 897
4003_6190	Pulse-Out n Enable register (PDB0_POEN)	32	R/W	0000_0000h	38.3.11/ 898
4003_6194	Pulse-Out n Delay register (PDB0_PO0DLY)	32	R/W	0000_0000h	38.3.12/ 898

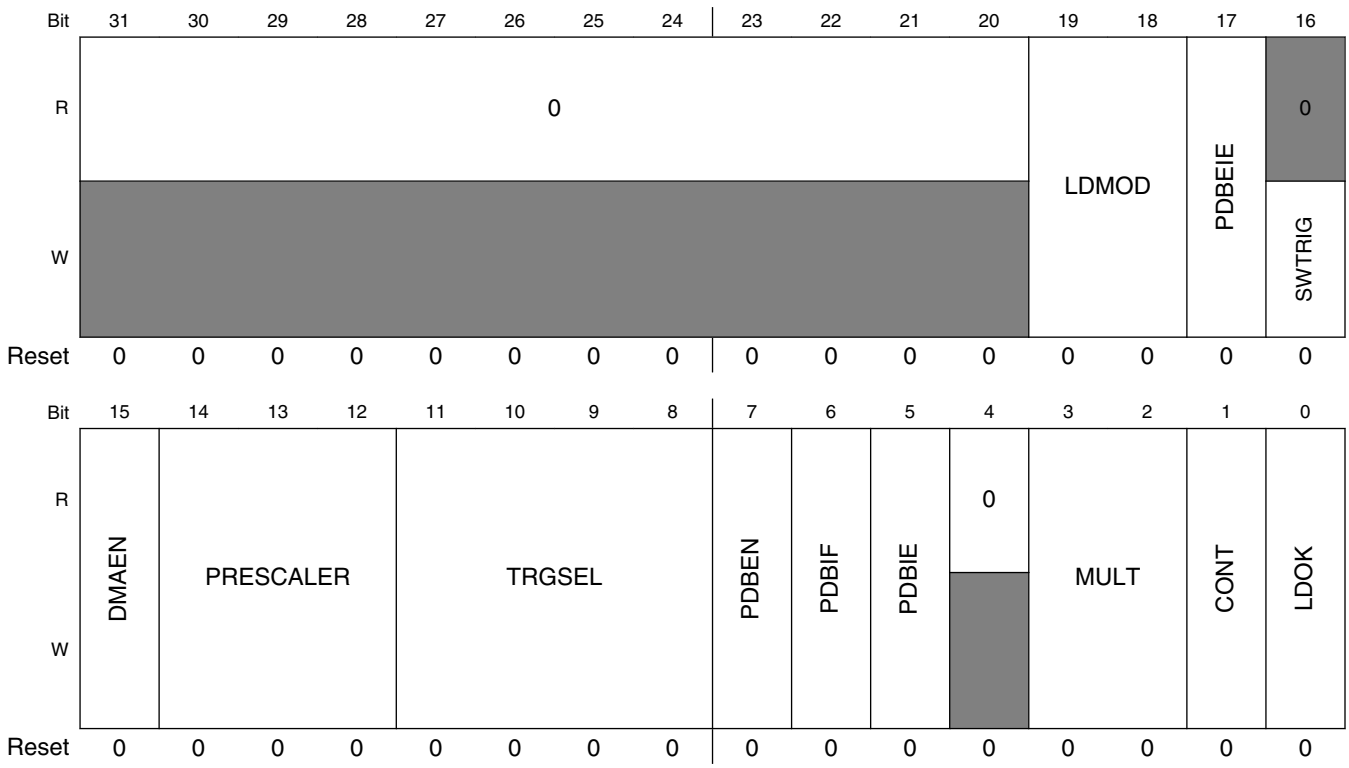
Table continues on the next page...

PDB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_6198	Pulse-Out n Delay register (PDB0_PO1DLY)	32	R/W	0000_0000h	38.3.12/ 898

38.3.1 Status and Control register (PDBx_SC)

Address: Base address + 0h offset



PDBx_SC field descriptions

Field	Description
31–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–18 LDMOD	Load Mode Select Selects the mode to load the MOD, IDLY, CH*n*DLY*m*, INT*x*, and PO*y*DLY registers, after 1 is written to LDOK. 00 The internal registers are loaded with the values from their buffers, immediately after 1 is written to LDOK. 01 The internal registers are loaded with the values from their buffers when the PDB counter (CNT) = MOD + 1 CNT delay elapsed, after 1 is written to LDOK.

Table continues on the next page...

PDBx_SC field descriptions (continued)

Field	Description
	10 The internal registers are loaded with the values from their buffers when a trigger input event is detected, after 1 is written to LDOK. 11 The internal registers are loaded with the values from their buffers when either the PDB counter (CNT) = MOD + 1 CNT delay elapsed, or a trigger input event is detected, after 1 is written to LDOK.
17 PDBEIE	PDB Sequence Error Interrupt Enable Enables the PDB sequence error interrupt. When PDBEIE is set, any of the PDB channel sequence error flags generates a PDB sequence error interrupt. 0 PDB sequence error interrupt disabled. 1 PDB sequence error interrupt enabled.
16 SWTRIG	Software Trigger When PDB is enabled and the software trigger is selected as the trigger input source, writing 1 to SWTRIG resets and restarts the counter. Writing 0 to SWTRIG has no effect. Reading SWTRIG yields 0.
15 DMAEN	DMA Enable When DMA is enabled, the PDBIF flag generates a DMA request instead of an interrupt. 0 DMA disabled. 1 DMA enabled.
14–12 PRESCALER	Prescaler Divider Select Counting uses the peripheral clock divided by the product of a factor (selected by MULT field) and an integer factor (set by PRESCALAR field), or in other words, (peripheral clock)/(MULT x PRESCALAR). 000 Counting uses the peripheral clock divided by MULT (the multiplication factor). 001 Counting uses the peripheral clock divided by 2 x MULT (the multiplication factor). 010 Counting uses the peripheral clock divided by 4 x MULT (the multiplication factor). 011 Counting uses the peripheral clock divided by 8 x MULT (the multiplication factor). 100 Counting uses the peripheral clock divided by 16 x MULT (the multiplication factor). 101 Counting uses the peripheral clock divided by 32 x MULT (the multiplication factor). 110 Counting uses the peripheral clock divided by 64 x MULT (the multiplication factor). 111 Counting uses the peripheral clock divided by 128 x MULT (the multiplication factor).
11–8 TRGSEL	Trigger Input Source Select Selects the trigger input source for the PDB. The trigger input source can be internal or external (EXTRG pin), or the software trigger. Refer to chip configuration details for the actual PDB input trigger connections. 0000 Trigger-In 0 is selected. 0001 Trigger-In 1 is selected. 0010 Trigger-In 2 is selected. 0011 Trigger-In 3 is selected. 0100 Trigger-In 4 is selected. 0101 Trigger-In 5 is selected. 0110 Trigger-In 6 is selected. 0111 Trigger-In 7 is selected. 1000 Trigger-In 8 is selected. 1001 Trigger-In 9 is selected.

Table continues on the next page...

PDBx_SC field descriptions (continued)

Field	Description
	1010 Trigger-In 10 is selected. 1011 Trigger-In 11 is selected. 1100 Trigger-In 12 is selected. 1101 Trigger-In 13 is selected. 1110 Trigger-In 14 is selected. 1111 Software trigger is selected.
7 PDBEN	PDB Enable 0 PDB disabled. Counter is off. 1 PDB enabled.
6 PDBIF	PDB Interrupt Flag PDBIF is set when the counter value is equal to the IDLY register + 1. Write zero to clear PDBIF.
5 PDBIE	PDB Interrupt Enable Enables the PDB interrupt. When PDBIE is set and DMAEN is cleared, PDBIF generates a PDB interrupt. 0 PDB interrupt disabled. 1 PDB interrupt enabled.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 MULT	Multiplication Factor Select for Prescaler Selects the multiplication factor of the prescaler divider for the counter clock. 00 Multiplication factor is 1. 01 Multiplication factor is 10. 10 Multiplication factor is 20. 11 Multiplication factor is 40.
1 CONT	Continuous Mode Enable Enables the PDB operation in Continuous mode. 0 PDB operation in One-Shot mode 1 PDB operation in Continuous mode
0 LDOK	Load OK Writing 1 to LDOK bit updates the MOD, IDLY, CHnDLYm, DACINTx, and POyDLY registers with the values previously written to their internal buffers (and stored there). The new values of MOD, IDLY, CHnDLYm, DACINTx, and POyDLY registers will take effect according to the setting of the LDMOD field (Load Mode Select). Before 1 is written to the LDOK field, the values in the internal buffers of these registers are not effective, and new values cannot be written to the internal buffers until the existing values in the internal buffers are loaded into their corresponding registers. • LDOK can be written only when PDBEN is set, or LDOK can be written at the same time when PDBEN is written to 1. • LDOK is automatically cleared when the values in the internal buffers are loaded into the registers or when PDBEN bit (PDB Enable) is cleared. • Writing 0 to LDOK has no effect.

38.3.2 Modulus register (PDBx_MOD)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the SC[LDOK] bit.

Address: Base address + 4h 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																MOD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PDBx_MOD field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MOD	PDB Modulus Specifies the period of the counter. When the counter reaches this value, it will be reset back to zero. If the PDB is in Continuous mode, the count begins anew. Reading this field returns the value of the internal register that is effective for the current cycle of PDB.

38.3.3 Counter register (PDBx_CNT)

NOTE

Writing to this read-only register will generate a transfer error (and possibly a hard fault).

Address: Base address + 8h 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																CNT															
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	0	0	0

PDBx_CNT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PDBx_CNT field descriptions (continued)

Field	Description
CNT	PDB Counter Contains the current value of the counter.

38.3.4 Interrupt Delay register (PDBx_IDLY)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the SC[LDOK] bit.

Address: Base address + Ch 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																IDLY															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PDBx_IDLY field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
IDLY	PDB Interrupt Delay Specifies the delay value to schedule the PDB interrupt. It can be used to schedule an independent interrupt at some point in the PDB cycle. If enabled, a PDB interrupt is generated, when the counter is equal to the IDLY. Reading this field returns the value of internal register that is effective for the current cycle of the PDB.

38.3.5 Channel n Control register 1 (PDBx_CH*n*C1)

Each PDB channel has one control register, CHnC1. The bits in this register control the functionality of each PDB channel operation.

Address: Base address + 10h offset + (40d × i), where i=0d to 1d

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								BB								TOS								EN							
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	0	0	0

PDBx_CHnC1 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–16 BB	PDB Channel Pre-Trigger Back-to-Back Operation Enable These bits enable the PDB ADC pre-trigger operation as back-to-back mode. Only lower M pre-trigger bits are implemented in this MCU. Back-to-back operation enables the ADC conversions complete to trigger the next PDB channel pre-trigger and trigger output, so that the ADC conversions can be triggered on next set of configuration and results registers. Application code must only enable the back-to-back operation of the PDB pre-triggers at the leading of the back-to-back connection chain. 0 PDB channel's corresponding pre-trigger back-to-back operation disabled. 1 PDB channel's corresponding pre-trigger back-to-back operation enabled.
15–8 TOS	PDB Channel Pre-Trigger Output Select Selects the PDB ADC pre-trigger outputs. Only lower M pre-trigger fields are implemented in this MCU. 0 PDB channel's corresponding pre-trigger is in bypassed mode. The pre-trigger asserts one peripheral clock cycle after a rising edge is detected on selected trigger input source or software trigger is selected and SWTRIG is written with 1. 1 PDB channel's corresponding pre-trigger asserts when the counter reaches the channel delay register and one peripheral clock cycle after a rising edge is detected on selected trigger input source or software trigger is selected and SETRIG is written with 1.
EN	PDB Channel Pre-Trigger Enable These bits enable the PDB ADC pre-trigger outputs. Only lower M pre-trigger bits are implemented in this MCU. 0 PDB channel's corresponding pre-trigger disabled. 1 PDB channel's corresponding pre-trigger enabled.

38.3.6 Channel n Status register (PDBx_CHnS)

Address: Base address + 14h offset + (40d × i), where i=0d to 1d

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								CF								0								ERR							
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	0	0	0

PDBx_CHnS field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–16 CF	PDB Channel Flags The CF[*m*] field is set when the PDB counter (PDB_CNT) matches the value CHnDLY*m* + 1. Write 0 to clear CF.

Table continues on the next page...

PDBx_CHnS field descriptions (continued)

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ERR	PDB Channel Sequence Error Flags Only the lower M bits are implemented in this MCU. 0 Sequence error not detected on PDB channel's corresponding pre-trigger. 1 Sequence error detected on PDB channel's corresponding pre-trigger. ADCn block can be triggered for a conversion by one pre-trigger from PDB channel *n*. When one conversion, which is triggered by one of the pre-triggers from PDB channel *n*, is in progress, new trigger from PDB channel's corresponding pre-trigger m cannot be accepted by ADCn, and ERR[m] is set. Writing 0's to clear the sequence error flags.

38.3.7 Channel n Delay 0 register (PDBx_CHnDLY0)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the SC[LDOK] bit.

Address: Base address + 18h offset + (40d × i), where i=0d to 1d

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																DLY															
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	0	0	0

PDBx_CHnDLY0 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DLY	PDB Channel Delay Specifies the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading this field returns the value of internal register that is effective for the current PDB cycle.

38.3.8 Channel n Delay 1 register (PDB*x*_CH*n*DLY1)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the SC[LDOK] bit.

Address: Base address + 1Ch offset + (40d × i), where i=0d to 1d

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																DLY															
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	0	0	0

PDB*x*_CH*n*DLY1 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DLY	PDB Channel Delay These bits specify the delay value for the channel's corresponding pre-trigger. The pre-trigger asserts when the counter is equal to DLY. Reading these bits returns the value of internal register that is effective for the current PDB cycle.

38.3.9 DAC Interval Trigger n Control register (PDB*x*_DACINTC*n*)

Address: Base address + 150h offset + (8d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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														EXT	TOE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PDB*x*_DACINTC*n* field descriptions

Field	Description
31–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 EXT	DAC External Trigger Input Enable

Table continues on the next page...

PDBx_DACINTC*n* field descriptions (continued)

Field	Description
	This bit enables the external trigger for DAC interval counter. 0 DAC external trigger input disabled. DAC interval counter is reset and started counting when a rising edge is detected on selected trigger input source or software trigger is selected and SWTRIG is written with 1. 1 DAC external trigger input enabled. DAC interval counter is bypassed and DAC external trigger input triggers the DAC interval trigger.
0 TOE	DAC Interval Trigger Enable Enables the DAC interval trigger. 0 DAC interval trigger disabled. 1 DAC interval trigger enabled.

38.3.10 DAC Interval n register (PDBx_DACINT*n*)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the SC[LDOK] bit.

Address: Base address + 154h offset + (8d × i), where i=0d to 0d

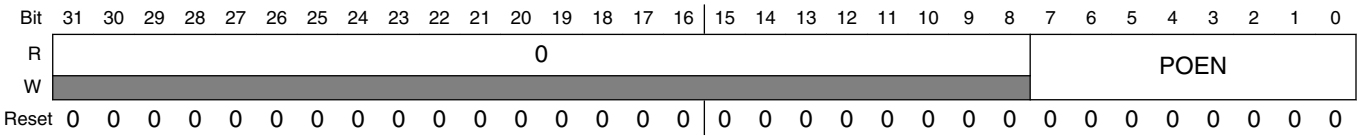
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																INT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PDBx_DACINT*n* field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
INT	DAC Interval These bits specify the interval value for DAC interval trigger. DAC interval trigger triggers DAC[1:0] update when the DAC interval counter is equal to the DACINT. Reading these bits returns the value of internal register that is effective for the current PDB cycle.

38.3.11 Pulse-Out n Enable register (PDB*x*_POEN)

Address: Base address + 190h offset



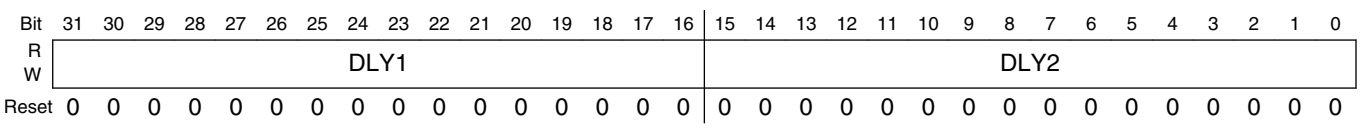
PDB*x*_POEN field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
POEN	PDB Pulse-Out Enable Enables the pulse output. Only lower 8 bits are implemented in this MCU. 0 PDB Pulse-Out disabled 1 PDB Pulse-Out enabled

38.3.12 Pulse-Out n Delay register (PDB*x*_PO*n*DLY)

Note: This register is internally buffered, and any values written to the register are written to its internal buffer instead; in other words, the internal device bus does not write directly to this register. The value in this register's internal buffer is loaded into this register only after "1" is written to the SC[LDOK] bit.

Address: Base address + 194h offset + (4d × i), where i=0d to 1d



PDB*x*_PO*n*DLY field descriptions

Field	Description
31–16 DLY1	PDB Pulse-Out Delay 1 Specifies the delay 1 value for the PDB Pulse-Out. Pulse-Out goes high when the PDB counter is equal to the DLY1. Reading this field returns the value of internal register that is effective for the current PDB cycle.
DLY2	PDB Pulse-Out Delay 2 Specifies the delay 2 value for the PDB Pulse-Out. Pulse-Out goes low when the PDB counter is equal to the DLY2. Reading this field returns the value of internal register that is effective for the current PDB cycle.

38.4 Functional description

38.4.1 PDB pre-trigger and trigger outputs

The PDB contains a counter whose output is compared to several different digital values. If the PDB is enabled, then a trigger input event will reset the counter and make it start to count. A trigger input event is defined as a rising edge being detected on a selected trigger input source, or if a software trigger is selected and SC[SWTRIG] is written with 1. For each channel, a delay *m* determines the time between assertion of the trigger input event to the time at which changes in the pre-trigger *m* output signal are started. The time is defined as:

- Trigger input event to pre-trigger *m* = (prescaler × multiplication factor × delay *m*) + 2 peripheral clock cycles
- Add 1 additional peripheral clock cycle to determine the time when the channel trigger output changes.

Each channel is associated with 1 ADC block. PDB channel n pre-trigger outputs 0 to *M*; each pre-trigger output is connected to ADC hardware trigger select and hardware trigger inputs. The pre-triggers are used to precondition the ADC block before the actual trigger occurs. When the ADC receives the rising edge of the trigger, the ADC will start the conversion according to the precondition determined by the pre-triggers. The ADC contains *M* sets of configuration and result registers, allowing it to alternate conversions between *M* different analog sources (like a ping-pong game). The pre-trigger outputs are used to specify which signal will be sampled next. When a pre-trigger *m* is asserted, the ADC conversion is triggered with set *m* of the configuration and result registers.

The waveforms shown in the following diagram show the pre-trigger and trigger outputs of PDB channel *n*. The delays can be independently set using the CH*n*DLY*m* registers, and the pre-triggers can be enabled or disabled in CH*n*C1[EN[*m*]].

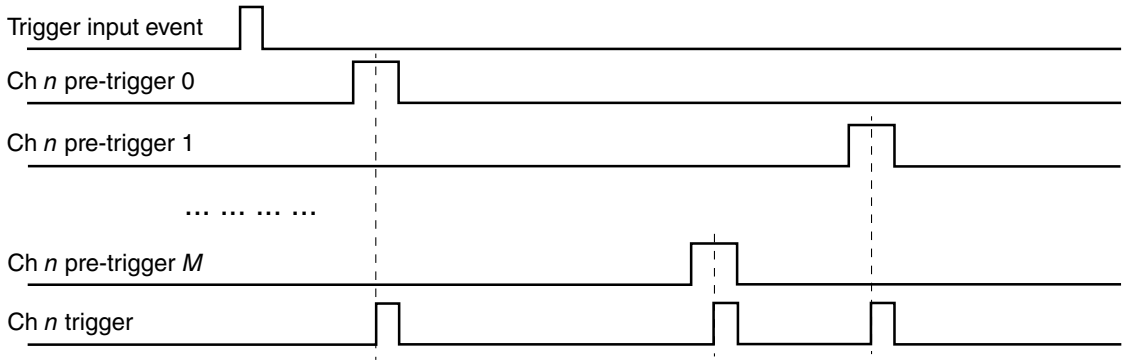


Figure 38-2. Pre-trigger and trigger outputs

The delay in CH*n*DLY*m* register can be optionally bypassed, if CH*n*C1[TOS[*m*]] is cleared. In this case, when the trigger input event occurs, the pre-trigger *m* is asserted after 2 peripheral clock cycles.

The PDB can be configured for back-to-back operation. Back-to-back operation enables the ADC conversion completions to trigger the next PDB channel pre-trigger and trigger outputs, so that the ADC conversions can be triggered on the next set of configuration and results registers. When back-to-back operation is enabled by setting CH*n*C1[BB[*m*]], then the delay *m* is ignored and the pre-trigger *m* is asserted 2 peripheral cycles after the acknowledgment *m* is received. The acknowledgment connections in this MCU are described in Back-to-back acknowledgment connections.

When a pre-trigger from a PDB channel *n* is asserted, the associated lock of the pre-trigger becomes active. The associated lock is released by the rising edge of the corresponding ADC*n*SC1[COCO]; the ADC*n*SC1[COCO] should be cleared after the conversion result is read, so that the next rising edge of ADC*n*SC1[COCO] can be generated to clear the lock later. The lock becomes inactive when:

- the rising edge of corresponding ADC*n*SC1[COCO] occurs,
- or the corresponding PDB pre-trigger is disabled,
- or the PDB is disabled

The channel *n* trigger output is suppressed when any of the locks of the pre-triggers in channel *n* is active. If a new pre-trigger *m* asserts when there is active lock in the PDB channel *n*, then a register flag bit CH*n*S[ERR[*m*]] (associated with the pre-trigger *m*) is set. If SC[PDBEIE] is set, then the sequence error interrupt is generated. A sequence error typically happens because the delay *m* is set too short and the pre-trigger *m* asserts before the previously triggered ADC conversion finishes.

When the PDB counter reaches the value (CNT + 1), the SC[PDBIF] flag is set. A PDB interrupt can be generated if SC[PDBIE] is set and SC[DMAEN] is cleared. If SC[DMAEN] is set, then the PDB requests a DMA transfer when the SC[PDBIF] flag is set.

The modulus value in the MOD register is used to reset the counter back to zero at the end of the count. If SC[CONT] is set, then the counter will then resume a new count; otherwise, the counter operation will stop until the next trigger input event occurs.

38.4.2 PDB trigger input source selection

The PDB has up to 15 trigger input sources, namely Trigger-In 0 to Trigger-In 14. They are connected to on-chip or off-chip event sources. The PDB can be triggered by software through SC[SWTRIG].

For the trigger input sources implemented in this MCU, see chip configuration information.

38.4.3 DAC interval trigger outputs

PDB can generate the interval triggers for DACs to update their outputs periodically. DAC interval counter *x* is reset and started when a trigger input event occurs if DACINTC*x*[EXT] is cleared. When the interval counter *x* is equal to the value set in DACINT*x* register, the DAC interval trigger *x* output generates a pulse of one peripheral clock cycle width to update the DAC*x*. If DACINTC*x*[EXT] is set, the DAC interval counter is bypassed and the interval trigger output *x* generates a pulse following the detection of a rising edge on the DAC external trigger input. The counter and interval trigger can be disabled by clearing the DACINTC*x*[TOE].

DAC interval counters are also reset when the PDB counter reaches the MOD register value; therefore, when the PDB counter rolls over to zero, the DAC interval counters starts anew.

The DAC interval trigger pulse and the ADC pre-trigger/trigger pulses together allow precise timing of DAC updates and ADC measurements. This is outlined in the typical use case described in the following diagram.

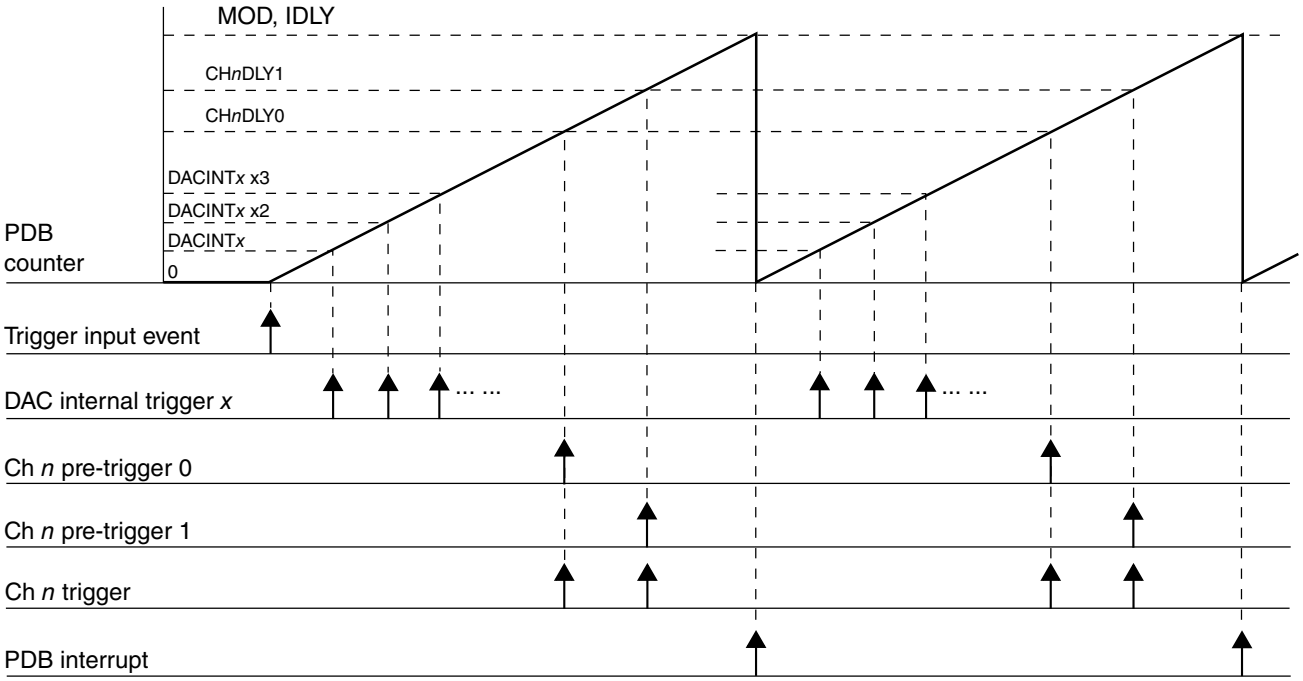


Figure 38-3. PDB ADC triggers and DAC interval triggers use case

NOTE

Because the DAC interval counters share the prescaler with PDB counter, PDB must be enabled if the DAC interval trigger outputs are used in the applications.

38.4.4 Pulse-Out's

PDB can generate pulse outputs of configurable width.

- When the PDB counter reaches the value set in PO*y*DLY[DLY1], then the Pulse-Out goes high.
- When the PDB counter reaches PO*y*DLY[DLY2], then it goes low.

PO*y*DLY[DLY2] can be set either greater or less than PO*y*DLY[DLY1].

ADC pre-trigger/trigger outputs and Pulse-Out generation have the same time base, because they both share the PDB counter. The pulse-out connections implemented in this MCU are described in the device's chip configuration details.

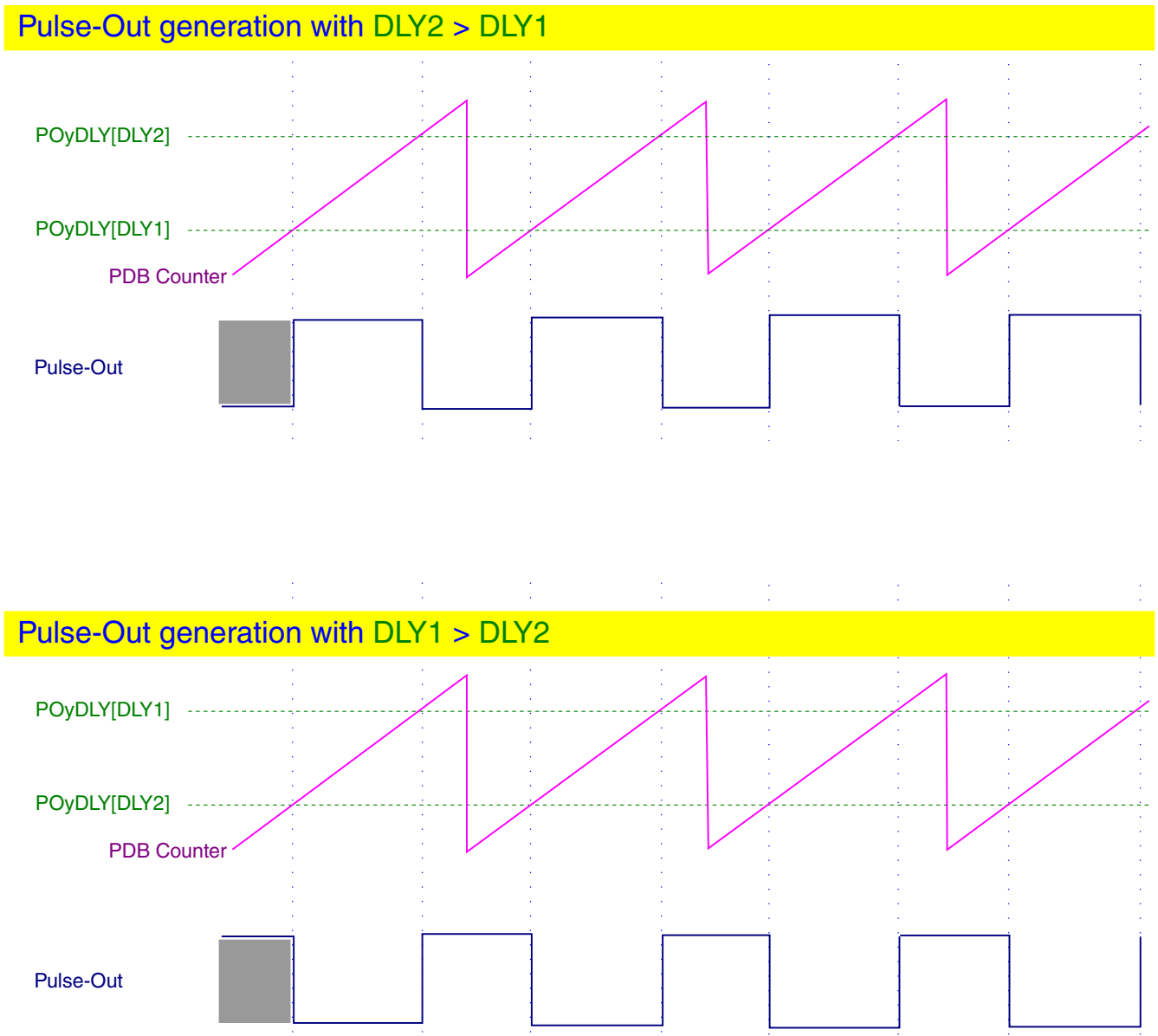


Figure 38-4. How Pulse Out is generated

38.4.5 Updating the delay registers

The following registers control the timing of the PDB operation; and in some of the applications, they may need to become effective at the same time.

- PDB Modulus register (MOD)
- PDB Interrupt Delay register (IDLY)
- PDB Channel *n* Delay *m* register (CH*n*DLY*m*)

- DAC Interval *x* register (DACINT*x*)
- PDB Pulse-Out *y* Delay register (PO*y*DLY)

The internal registers of them are buffered and any values written to them are written first to their buffers. The circumstances that cause their internal registers to be updated with the values from the buffers are summarized in the next table.

Table 38-2. When delay registers are updated

SC[LDMOD]	Update to the delay registers
00	The internal registers are loaded with the values from their buffers immediately after 1 is written to SC[LDOK].
01	The PDB counter reaches PDB_MOD[MOD] + 1 value, after 1 is written to SC[LDOK].
10	A trigger input event is detected after 1 is written to SC[LDOK].
11	Either the PDB counter reaches PDB_MOD[MOD] + 1 value or a trigger input event is detected, after 1 is written to SC[LDOK].

After 1 is written to SC[LDOK], the buffers cannot be written until the values in buffers are loaded into their internal registers. SC[LDOK] is self-cleared when the internal registers are loaded, so the application code can read it to determine the updates to the internal registers.

The following diagrams show the cases of the internal registers being updated with SC[LDMOD] is 00 and x1.

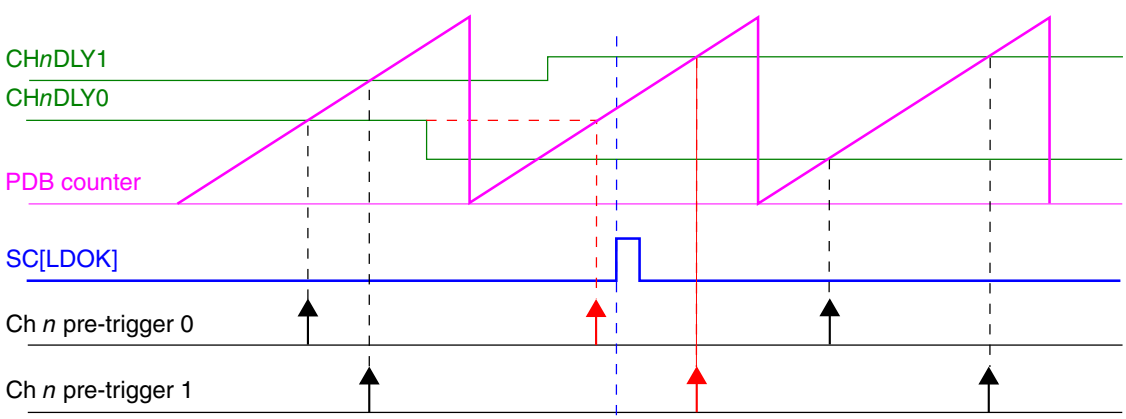


Figure 38-5. Registers update with SC[LDMOD] = 00

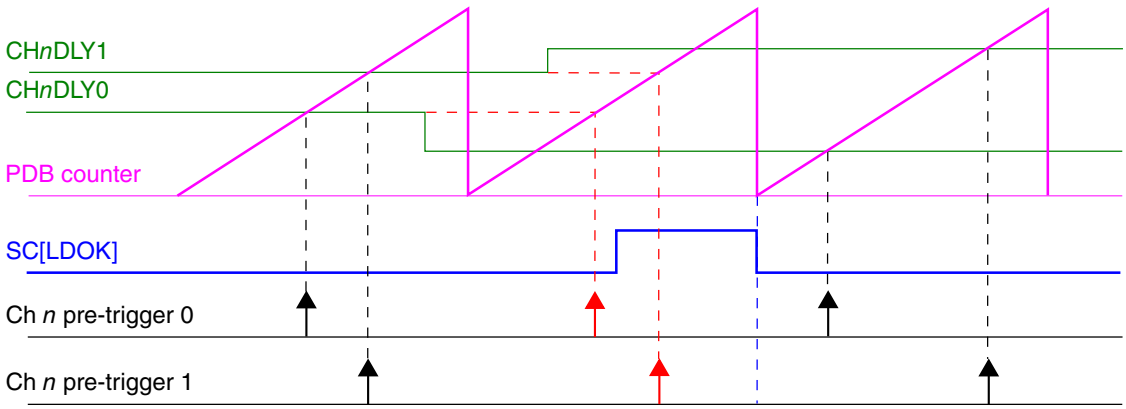


Figure 38-6. Registers update with SC[LDMOD] = x1

38.4.6 Interrupts

PDB can generate two interrupts: PDB interrupt and PDB sequence error interrupt. The following table summarizes the interrupts.

Table 38-3. PDB interrupt summary

Interrupt	Flags	Enable bit
PDB Interrupt	SC[PDBIF]	SC[PDBIE] = 1 and SC[DMAEN] = 0
PDB Sequence Error Interrupt	CH*n*S[ERR*m*]	SC[PDBEIE] = 1

38.4.7 DMA

If SC[DMAEN] is set, PDB can generate a DMA transfer request when SC[PDBIF] is set. When DMA is enabled, the PDB interrupt is not issued.

38.5 Application information

38.5.1 Impact of using the prescaler and multiplication factor on timing resolution

Use of prescaler and multiplication factor greater than 1 limits the count/delay accuracy in terms of peripheral clock cycles (to the modulus of the prescaler X multiplication factor). If the multiplication factor is set to 1 and the prescaler is set to 2 then the only values of total peripheral clocks that can be detected are even values; if prescaler 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 applications need a really long delay value and use a prescaler set to 128, then the resolution would be limited to 128 peripheral clock cycles.

Therefore, use the lowest possible prescaler and multiplication factor for a given application.

Chapter 39
CAN (FlexCAN)

39.1 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 ISO 11898-1 standard and CAN 2.0 B protocol specifications. 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 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.

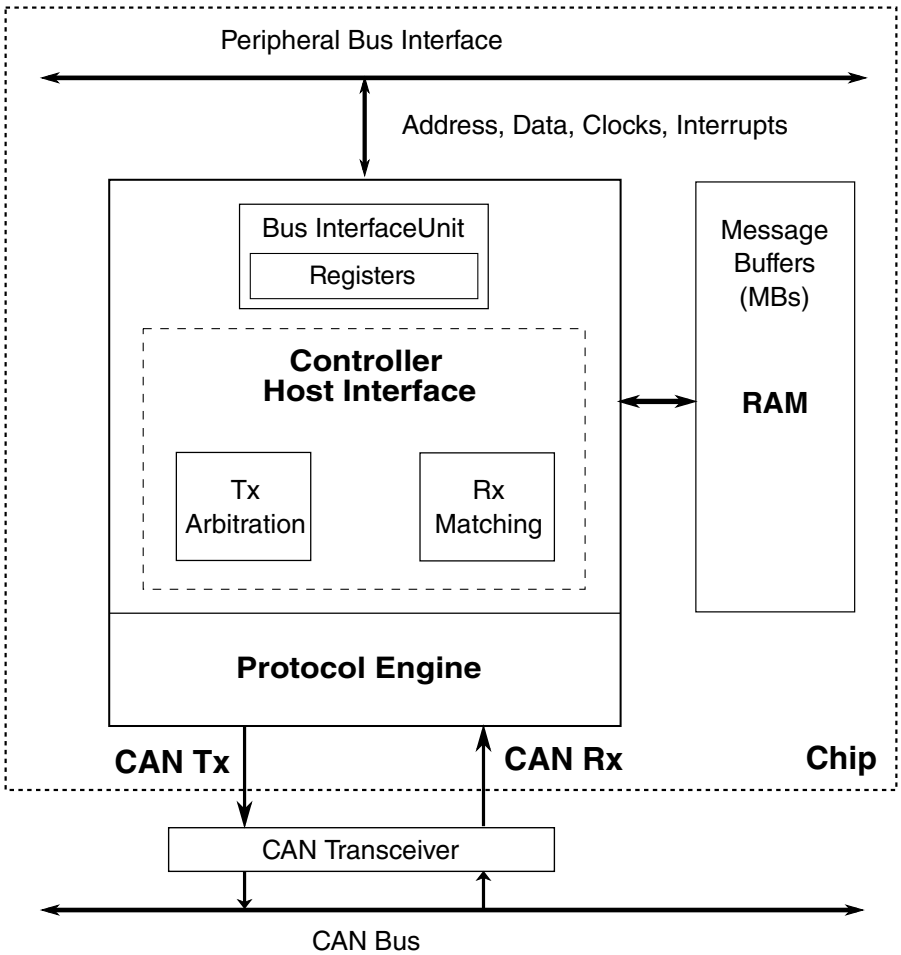


Figure 39-1. FlexCAN block diagram

39.1.1 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, the CAN 2.0 version B protocol, 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 chip.

The 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, DMA and test signals are accessed through the BIU.

39.1.2 FlexCAN module features

The FlexCAN module includes these distinctive features:

- Full implementation of the CAN protocol specification, Version 2.0 B
  - Standard data frames
  - Extended data 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 with DMA support
- Transmission abort capability
- Flexible message buffers (MBs), totaling 16 message buffers of 8 bytes data length each, configurable as Rx or Tx

- Programmable clock source to the CAN Protocol Interface, either peripheral clock or oscillator clock
- RAM not used by reception or transmission structures 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
- 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

39.1.3 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 chip 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.

- 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.

- 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.

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 chip 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 chip 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 chip 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.

39.2 FlexCAN signal descriptions

The FlexCAN module has two I/O signals connected to the external chip pins. These signals are summarized in the following table and described in more detail in the next subsections.

Table 39-1. FlexCAN signal descriptions

Signal	Description	I/O
CAN Rx	CAN Receive Pin	Input
CAN Tx	CAN Transmit Pin	Output

39.2.1 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.

39.2.2 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.

39.3 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 chip.

39.3.1 FlexCAN memory mapping

The memory map for the 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 offset 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 39-2.

Table 39-2. Register access and reset information

Register	Access type	Affected by hard reset	Affected by soft reset
Module Configuration Register (CAN_MCR)	S	Yes	Yes
Control 1 register (CAN_CTRL1)	S/U	Yes	No
Free Running Timer register (CAN_TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask register (CAN_RXMGMASK)	S/U	No	No
Rx Buffer 14 Mask register (CAN_RX14MASK)	S/U	No	No
Rx Buffer 15 Mask register (CAN_RX15MASK)	S/U	No	No
Error Counter Register (CAN_ECR)	S/U	Yes	Yes
Error and Status 1 Register (CAN_ESR1)	S/U	Yes	Yes
Interrupt Masks 1 register (CAN_IMASK1)	S/U	Yes	Yes
Interrupt Flags 1 register (CAN_IFLAG1)	S/U	Yes	Yes
Control 2 Register (CAN_CTRL2)	S/U	Yes	No
Error and Status 2 Register (CAN_ESR2)	S/U	Yes	Yes
CRC Register (CAN_CRCR)	S/U	Yes	Yes

Table continues on the next page...

Table 39-2. Register access and reset information (continued)

Register	Access type	Affected by hard reset	Affected by soft reset
Rx FIFO Global Mask register (CAN_RXFGMASK)	S/U	No	No
Rx FIFO Information Register (CAN_RXFIR)	S/U	No	No
CAN Bit Timing Register (CAN_CBT)	S/U	Yes	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.

The table below shows the FlexCAN memory map.

The address range from offset 0x80 to 0x17F allocates the sixteen 128-bit Message Buffers (MBs).

The memory maps for the message buffers are in FlexCAN message buffer memory map.

CAN memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_4000	Module Configuration Register (CAN0_MCR)	32	R/W	See section	39.3.2/916
4002_4004	Control 1 register (CAN0_CTRL1)	32	R/W	0000_0000h	39.3.3/921
4002_4008	Free Running Timer (CAN0_TIMER)	32	R/W	0000_0000h	39.3.4/925
4002_4010	Rx Mailboxes Global Mask Register (CAN0_RXMGMASK)	32	R/W	Undefined	39.3.5/926
4002_4014	Rx 14 Mask register (CAN0_RX14MASK)	32	R/W	Undefined	39.3.6/927
4002_4018	Rx 15 Mask register (CAN0_RX15MASK)	32	R/W	Undefined	39.3.7/928
4002_401C	Error Counter (CAN0_ECR)	32	R/W	0000_0000h	39.3.8/928
4002_4020	Error and Status 1 register (CAN0_ESR1)	32	R/W	See section	39.3.9/930
4002_4028	Interrupt Masks 1 register (CAN0_IMASK1)	32	R/W	0000_0000h	39.3.10/ 936
4002_4030	Interrupt Flags 1 register (CAN0_IFLAG1)	32	R/W	0000_0000h	39.3.11/ 936
4002_4034	Control 2 register (CAN0_CTRL2)	32	R/W	See section	39.3.12/ 939
4002_4038	Error and Status 2 register (CAN0_ESR2)	32	R/W	0000_0000h	39.3.13/ 942
4002_4044	CRC Register (CAN0_CRCR)	32	R	0000_0000h	39.3.14/ 944
4002_4048	Rx FIFO Global Mask register (CAN0_RXFGMASK)	32	R/W	Undefined	39.3.15/ 944

Table continues on the next page...

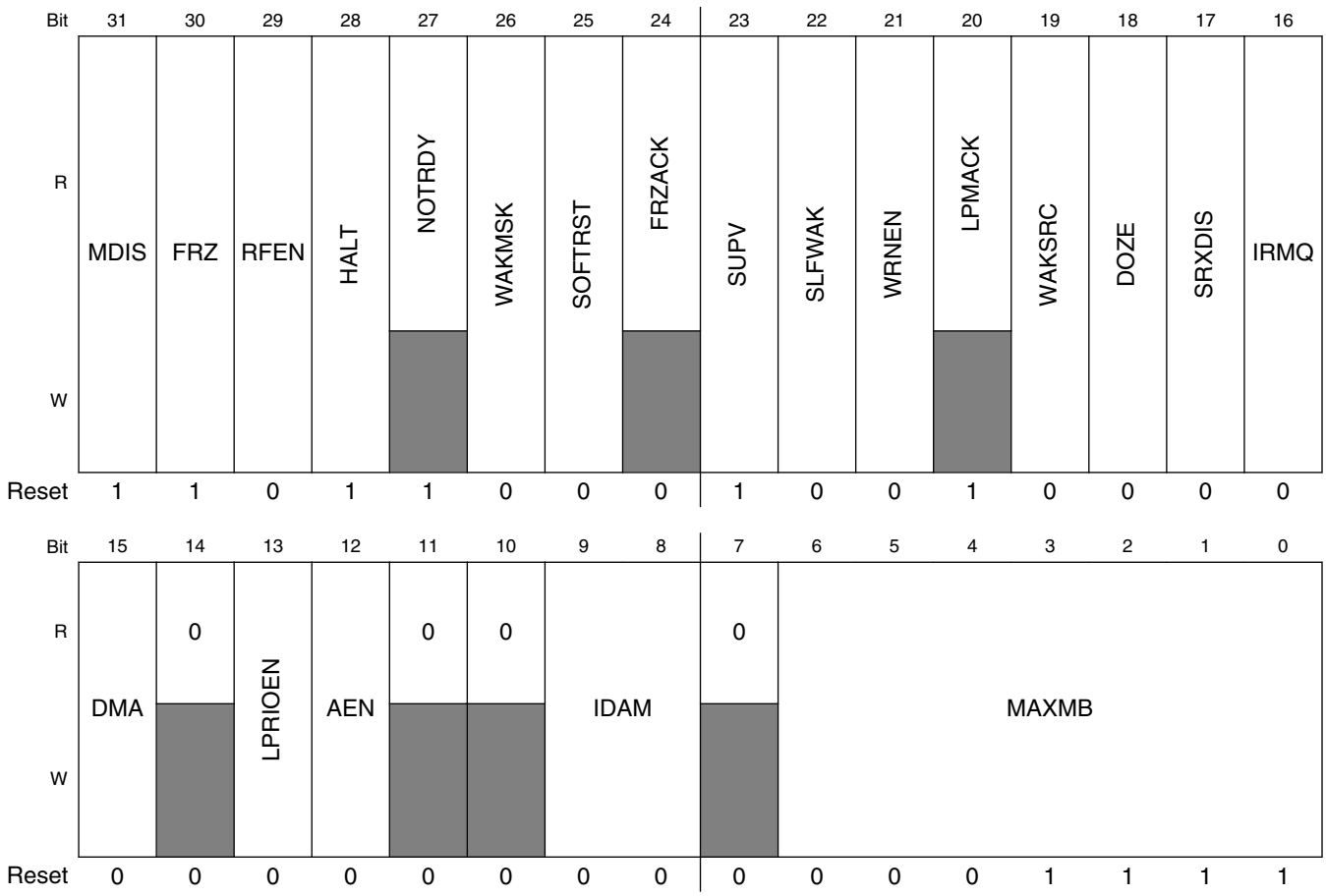
CAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_404C	Rx FIFO Information Register (CAN0_RXFIR)	32	R	Undefined	39.3.16/ 945
4002_4050	CAN Bit Timing Register (CAN0_CBT)	32	R/W	See section	39.3.17/ 946
4002_4880	Rx Individual Mask Registers (CAN0_RXIMR0)	32	R/W	Undefined	39.3.18/ 948
4002_4884	Rx Individual Mask Registers (CAN0_RXIMR1)	32	R/W	Undefined	39.3.18/ 948
4002_4888	Rx Individual Mask Registers (CAN0_RXIMR2)	32	R/W	Undefined	39.3.18/ 948
4002_488C	Rx Individual Mask Registers (CAN0_RXIMR3)	32	R/W	Undefined	39.3.18/ 948
4002_4890	Rx Individual Mask Registers (CAN0_RXIMR4)	32	R/W	Undefined	39.3.18/ 948
4002_4894	Rx Individual Mask Registers (CAN0_RXIMR5)	32	R/W	Undefined	39.3.18/ 948
4002_4898	Rx Individual Mask Registers (CAN0_RXIMR6)	32	R/W	Undefined	39.3.18/ 948
4002_489C	Rx Individual Mask Registers (CAN0_RXIMR7)	32	R/W	Undefined	39.3.18/ 948
4002_48A0	Rx Individual Mask Registers (CAN0_RXIMR8)	32	R/W	Undefined	39.3.18/ 948
4002_48A4	Rx Individual Mask Registers (CAN0_RXIMR9)	32	R/W	Undefined	39.3.18/ 948
4002_48A8	Rx Individual Mask Registers (CAN0_RXIMR10)	32	R/W	Undefined	39.3.18/ 948
4002_48AC	Rx Individual Mask Registers (CAN0_RXIMR11)	32	R/W	Undefined	39.3.18/ 948
4002_48B0	Rx Individual Mask Registers (CAN0_RXIMR12)	32	R/W	Undefined	39.3.18/ 948
4002_48B4	Rx Individual Mask Registers (CAN0_RXIMR13)	32	R/W	Undefined	39.3.18/ 948
4002_48B8	Rx Individual Mask Registers (CAN0_RXIMR14)	32	R/W	Undefined	39.3.18/ 948
4002_48BC	Rx Individual Mask Registers (CAN0_RXIMR15)	32	R/W	Undefined	39.3.18/ 948

39.3.2 Module Configuration Register (CANx_MCR)

This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

Address: 4002_4000h base + 0h offset = 4002_4000h



CANx_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 bit is 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 CAN_MCR Register is set or when Debug mode is requested at chip 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...

CANx_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 CAN_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" (see Arbitration and matching timing). This bit can be written in Freeze mode only because 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 the Control Registers CAN_CTRL1 and CAN_CTRL2. 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. This bit is not affected by soft reset. 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 under Self Wake Up mechanism. 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 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 chip level. Because 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. This bit is not affected by soft reset. 0 No reset request. 1 Resets the registers affected by soft reset.
24 FRZACK	Freeze Mode Acknowledge

Table continues on the next page...

CANx_MCR field descriptions (continued)

Field	Description
	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 after the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in a low power mode, then the FRZACK bit will be set only when the low-power mode is exited. See Section "Freeze Mode". This bit is not affected by soft reset. **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 be written only in Freeze mode because 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 1 (ESR1). 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 be written in Freeze mode only because 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 This read-only bit indicates that FlexCAN is in a low-power mode (Disable mode, Doze mode, Stop mode). A low-power mode cannot be entered until all current transmission or reception processes have finished,

Table continues on the next page...

CANx_MCR field descriptions (continued)

Field	Description
	so the CPU can poll the LPMACK bit to know when FlexCAN has actually entered low power mode. This bit is not affected by soft reset. **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 be written only in Freeze mode because 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 chip 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 be written only in Freeze mode because 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 CAN_RXMGMASK, CAN_RX14MASK, CAN_RX15MASK and CAN_RXFGMASK. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Individual Rx masking and queue feature are disabled. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY. 1 Individual Rx masking and queue feature are enabled.
15 DMA	DMA Enable The DMA Enable bit controls whether the DMA feature is enabled or not. The DMA feature can only be used in Rx FIFO, consequently the bit CAN_MCR[RFEN] must be asserted. When DMA and RFEN are set, the CAN_IFLAG1[BUF5I] generates the DMA request and no RX FIFO interrupt is generated. This bit can be written in Freeze mode only as it is blocked by hardware in other modes. 0 DMA feature for RX FIFO disabled. 1 DMA feature for RX FIFO enabled.

Table continues on the next page...

CANx_MCR field descriptions (continued)

Field	Description
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 with legacy applications. 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 be written only in Freeze mode because it is blocked by hardware in other modes. 0 Local Priority disabled. 1 Local Priority enabled.
12 AEN	Abort Enable When asserted, this bit 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 be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** When CAN_MCR[AEN] is asserted, only the abort mechanism (see 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 CAN_MCR[AEN] is asserted. 0 Abort disabled. 1 Abort enabled.
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 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 be written only in Freeze mode because 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 a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Number of the last MB = MAXMB **NOTE:** MAXMB must be programmed with a value smaller than or equal to the number of available Message Buffers.

Table continues on the next page...

CANx_MCR field descriptions (continued)

Field	Description
	Additionally, the definition of MAXMB value must take into account the region of MBs occupied by Rx FIFO and its ID filters table space defined by RFFN bit in CAN_CTRL2 register. 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" (see Arbitration and matching timing).

39.3.3 Control 1 register (CANx_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.

The CAN bit timing variables (PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW) can also be configured in CAN_CBT register, which extends the range of all these variables. If CAN_CBT[BTF] is set, PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW fields of CAN_CTRL1 become read only.

The contents of this register are not affected by soft reset.

NOTE

The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.

Address: 4002_4000h base + 4h offset = 4002_4004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRESDIV								RJW		PSEG1			PSEG2		
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	BOFFMSK	ERRMSK	CLKSRC	LPB	TWRNMSK	RWRNMSK	Reserved	Reserved	SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_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 be written only in Freeze mode because 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 be written only in Freeze mode because 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 Segment 1 in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta.
18–16 PSEG2	Phase Segment 2 This 3-bit field defines the length of Phase Segment 2 in the bit time. The valid programmable values are 1–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta.
15 BOFFMSK	Bus Off Interrupt Mask This bit provides a mask for the Bus Off Interrupt BOFFINT in CAN_ESR1 register. 0 Bus Off interrupt disabled. 1 Bus Off interrupt enabled.
14 ERRMSK	Error Interrupt Mask This bit provides a mask for the Error Interrupt ERRINT in the CAN_ESR1 register. 0 Error interrupt disabled. 1 Error interrupt enabled.
13 CLKSRC	CAN Engine Clock Source

Table continues on the next page...

CANx_CTRL1 field descriptions (continued)

Field	Description
	This bit selects the clock source to the CAN Protocol Engine (PE) to be either the peripheral clock or the 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 be written only in Disable mode because it is blocked by hardware in other modes. See 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 be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** In this mode, the CAN_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 1 (ESR1). This bit is read as zero when CAN_MCR[WRNEN] bit is negated. This bit can be written only if CAN_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 1 (ESR1). This bit is read as zero when CAN_MCR[WRNEN] bit is negated. This bit can be written only if CAN_MCR[WRNEN] bit is asserted. 0 Rx Warning Interrupt disabled. 1 Rx Warning Interrupt enabled.
9 Reserved	This field is reserved.
8 Reserved	This field is reserved.
7 SMP	CAN Bit Sampling This bit defines the sampling mode of CAN bits at the Rx input. It can be written in Freeze mode only because it is blocked by hardware in other modes. **NOTE:** For proper operation, to assert SMP it is necessary to guarantee a minimum value of 2 TQs in CAN_CTRL1[PSEG1] (or CAN_CBT[EPSEG1]).

Table continues on the next page...

CANx_CTRL1 field descriptions (continued)

Field	Description
	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 be effective only 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. 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, that is, global network time. If the RFEN bit in CAN_MCR is set (Rx FIFO enabled), the first available Mailbox, according to CAN_CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written in Freeze mode only because 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 CAN_MCR[LPRIOEN] bit does not affect the priority arbitration. This bit can be written in Freeze mode only because 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 described in CAN_ECR register 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 receive error counter (RXERRCNT) in CAN_ECR register, as if it was trying to acknowledge the message. Listen-Only mode is acknowledged by the state of CAN_ESR1[FLTCONF] field indicating Passive Error. There can be some delay between the Listen-Only mode request and acknowledge. This bit can be written in Freeze mode only because 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

Table continues on the next page...

CANx_CTRL1 field descriptions (continued)

Field	Description
	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 be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.

39.3.4 Free Running Timer (CANx_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 incremented by the CAN 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.

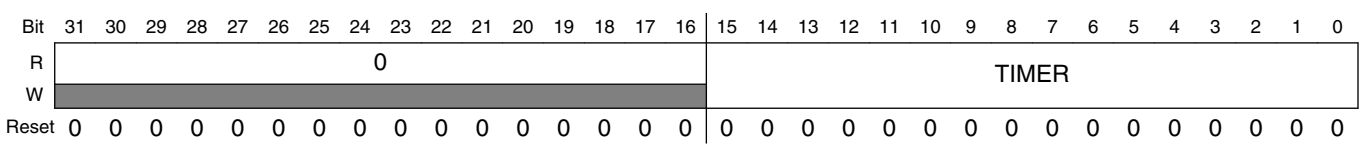
The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CAN_CTRL1[TSYN] is asserted, the Timer is reset whenever a message is received in the first available Mailbox, according to CAN_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 "Mailbox Lock Mechanism".

Address: 4002_4000h base + 8h offset = 4002_4008h



CANx_TIMER field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CANx_TIMER field descriptions (continued)

Field	Description
TIMER	Timer Value Contains the free-running counter value.

39.3.5 Rx Mailboxes Global Mask Register (CANx_RXMGMASK)

This register is located in RAM.

RXMGMASK is provided for legacy application support.

- When the CAN_MCR[IRMQ] bit is negated, RXMGMASK is always in effect (the bits in the MG field will mask the Mailbox filter bits).
- When the CAN_MCR[IRMQ] bit is asserted, RXMGMASK has no effect (the bits in the MG field will not mask the Mailbox filter bits).

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: 4002_4000h base + 10h offset = 4002_4010h

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																															
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.

CANx_RXMGMASK field descriptions

Field	Description
MG	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]	CAN_CTRL2[RRS]	CAN_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]

CANx_RXMGMASK field descriptions (continued)

Field	Description

SMB[RTR] [1]	CAN_CTRL2[RRS]	CAN_CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	Reserved
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]

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 CAN_CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

39.3.6 Rx 14 Mask register (CANx_RX14MASK)

This register is located in RAM.

RX14MASK is provided for legacy application support. When the CAN_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: 4002_4000h base + 14h offset = 4002_4014h

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																															
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.

CANx_RX14MASK field descriptions

Field	Description
RX14M	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.

39.3.7 Rx 15 Mask register (CANx_RX15MASK)

This register is located in RAM.

RX15MASK is provided for legacy application support. When the CAN_MCR[IRMQ] bit is asserted, RX15MASK has no effect.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can be programmed only while the module is in Freeze mode because it is blocked by hardware in other modes.

Address: 4002_4000h base + 18h offset = 4002_4018h

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																															
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.

CANx_RX15MASK field descriptions

Field	Description
RX15M	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.

39.3.8 Error Counter (CANx_ECR)

This register has two 8-bit fields reflecting the value of the FlexCAN error counters:

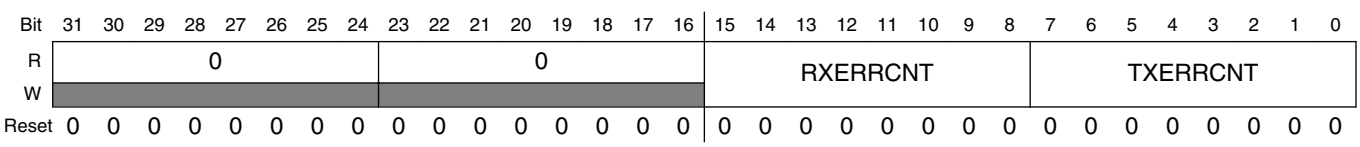
- Transmit Error Counter (TXERRCNT field)
- Receive Error Counter (RXERRCNT field)

The Fault Confinement State (FLTCONF field in Error and Status Register 1 - CAN_ESR1) is updated based on TXERRCNT and RXERRCNT counters. TXERRCNT and RXERRCNT counters can be written in Freeze mode only. The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module.

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: 4002_4000h base + 1Ch offset = 4002_401Ch



CANx_ECR field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CANx_ECR field descriptions (continued)

Field	Description
15–8 RXERRCNT	Receive Error Counter Receive Error Counter for all errors detected in received messages. The RXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.
TXERRCNT	Transmit Error Counter Transmit Error Counter for all errors detected in transmitted messages. The TXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.

39.3.9 Error and Status 1 register (CANx_ESR1)

This register reports various error conditions detected in the reception and transmission of a CAN frame, some general status of the device and it is the source of some interrupts to the CPU.

The reported error conditions are BIT1ERR, BIT0ERR, ACKKERR, CRCERR, FRMERR and STFERR.

An error detected in a single CAN frame may be reported by one or more error flags. Also, error reporting is cummulative in case more error events happen in the next frames while the CPU does not attempt to read this register.

TXWRN, RXWRN, IDLE, TX, FLTCONF, RX and SYNCH are status bits.

BOFFINT, BOFFDONEINT, ERRINT, WAKINT, TWRNINT and RWRNINT are interrupt bits. It is recommended the CPU to use the following procedure when servicing interrupt requests generated by these bits:

- Read this register to capture all error condition and status bits. This action clear the respective bits that were set since the last read access.
- Write 1 to clear the interrupt bit that has triggered the interrupt request.
- Write 1 to clear the ERR_OVR bit if it is set.

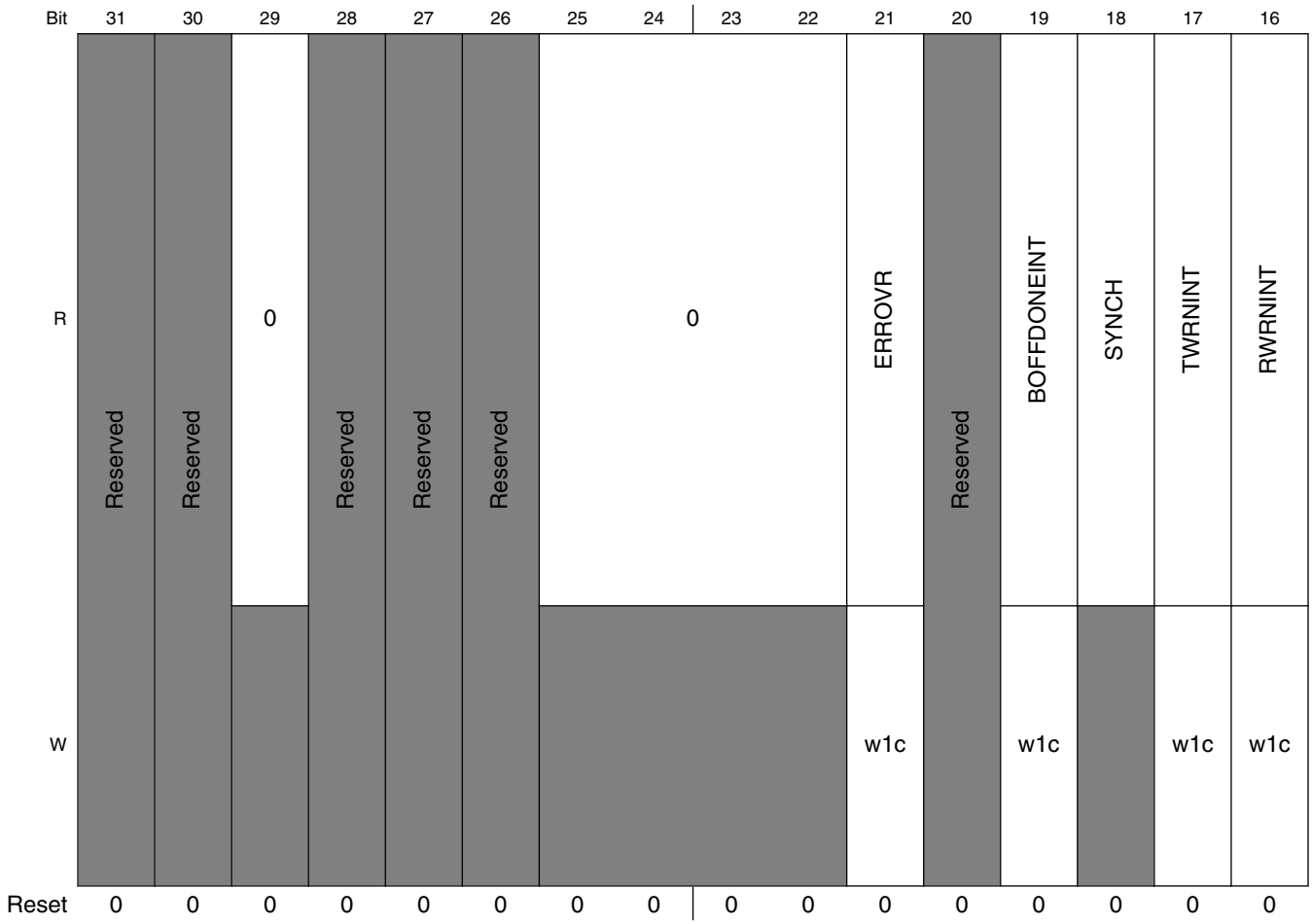
Starting from all error flags cleared, a first error event sets the ERRINT (provided the corresponding mask bit is asserted). If other error events in subsequent frames happen before the CPU to serve the interrupt request, the ERR_OVR bit is set to indicate that errors from different frames had accumulated.

SYNCH	IDLE	TX	RX	FlexCAN State
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle

Table continues on the next page...

1	0	1	0	Transmitting
1	0	0	1	Receiving

Address: 4002_4000h base + 20h offset = 4002_4020h



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

CANx_ESR1 field descriptions

Field	Description
31 Reserved	This field is reserved.
30 Reserved	This field is reserved.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved.
27 Reserved	This field is reserved.
26 Reserved	This field is reserved.
25–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 ERROVR	Error Overrun bit This bit indicates that an error condition occurred when any error flag is already set. This bit is cleared by writing it to 1.

Table continues on the next page...

CANx_ESR1 field descriptions (continued)

Field	Description
	0 Overrun has not occurred. 1 Overrun has occurred.
20 Reserved	This field is reserved.
19 BOFFDONEINT	Bus Off Done Interrupt This bit is set when the Tx Error Counter (TXERRCNT) has finished counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave Bus Off. If the corresponding mask bit in the Control 2 Register (BOFFDONEMSK) 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 has completed Bus Off process.
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 CAN_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 1 Register (CAN_CTRL1[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 CAN_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 1 Register (CAN_CTRL1[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.

Table continues on the next page...

CANx_ESR1 field descriptions (continued)

Field	Description
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, that is, a dominant bit has not been detected during the ACK SLOT. 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, that is, 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, that is, 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 by the receiver node. 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 and is affected by the value of TXERRCNT in CAN_ECR register only. 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 and is affected by the value of RXERRCNT in CAN_ECR register only. 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.

Table continues on the next page...

CANx_ESR1 field descriptions (continued)

Field	Description
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. If the LOM bit in the Control Register 1 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 CAN_ECR register by the CPU. It may be necessary up to one CAN bit time to get them coherent again. This bit field is affected by soft reset, but if the LOM bit is asserted, its reset value lasts just one CAN bit. After this time, FLTCONF reports "Error Passive". 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 1 (CAN_CTRL1[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 (BIT1ERR, BIT0ERR, ACKERR. CRCERR, FRMERR or STFERR) is set. If the corresponding mask bit CAN_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 under Self Wake Up mechanism: • Doze mode • Stop mode When a recessive-to-dominant transition is detected on the CAN bus and if the CAN_MCR[WAKMSK] bit is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When CAN_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.

Table continues on the next page...

CANx_ESR1 field descriptions (continued)

Field	Description
	0 No such occurrence. 1 Indicates a recessive to dominant transition was received on the CAN bus.

39.3.10 Interrupt Masks 1 register (CANx_IMASK1)

This register allows any number of a range of the 16 Message Buffer Interrupts to be enabled or disabled for MB15 to MB0. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG1 bit is set.

Address: 4002_4000h base + 28h offset = 4002_4028h

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	BUF31TO0M																															
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

CANx_IMASK1 field descriptions

Field	Description
BUF31TO0M	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB31 to MB0. **NOTE:** Setting or clearing a bit in the CAN_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.

39.3.11 Interrupt Flags 1 register (CANx_IFLAG1)

This register defines the flags for the 16 Message Buffer interrupts for MB15 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding CAN_IFLAG1 bit. If the corresponding CAN_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. There is an exception when DMA for Rx FIFO is enabled, as described below.

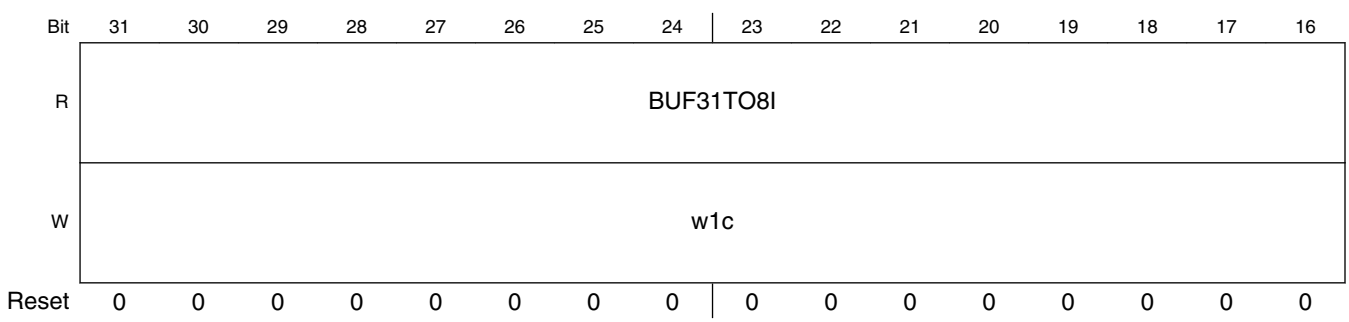
The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the bit CAN_MCR[RFEN] is set and the bit CAN_MCR[DMA] is negated, the function of the 8 least significant interrupt flags changes: BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, BUF0I is used to empty FIFO, and BUF4I to BUF1I bits are reserved.

Before enabling the CAN_MCR[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 CAN_MCR[RFEN] bit is negated, the FIFO flags must be cleared. The same care must be taken when an CAN_CTRL2[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.

When both the CAN_MCR[RFEN] and CAN_MCR[DMA] bits are asserted (DMA feature for Rx FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I - BUF0I) are changed to support the DMA operation. BUF7I and BUF6I are not used, as well as, BUF4I to BUF1I. BUF5I indicates operating condition of FIFO, and BUF0I is used to empty FIFO. Moreover, BUF5I does not generate a CPU interrupt, but generates a DMA request. IMASK1 bits in Rx FIFO region are not considered when bit CAN_MCR[DMA] is enabled. In addition the CPU must not clear the flag BUF5I when DMA is enabled. Before enabling the bit CAN_MCR[DMA], the CPU must service the IFLAGs asserted in the Rx FIFO region. When the bit CAN_MCR[DMA] is negated, the FIFO must be empty.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG1 bits whose MB value is greater than the CAN_MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 4002_4000h base + 30h offset = 4002_4030h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUF31TO8I								BUF7I	BUF6I	BUF5I	BUF4TO1I				BUF0I
W	w1c								w1c	w1c	w1c	w1c				w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CANx_IFLAG1 field descriptions

Field	Description
31–8 BUF31TO8I	Buffer MB_i Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB15 to MB8. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.
7 BUF7I	Buffer MB7 Interrupt Or "Rx FIFO Overflow" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF7I flag represents "Rx FIFO Overflow" when CAN_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 CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF6I flag represents "Rx FIFO Warning" when CAN_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.

Table continues on the next page...

CANx_IFLAG1 field descriptions (continued)

Field	Description
	NOTE: This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. When MCR[RFEN] is set (Rx FIFO enabled), the BUF5I flag represents "Frames available in Rx FIFO" and indicates that at least one frame is available to be read from the Rx FIFO. When the MCR[DMA] bit is enabled, this flag generates a DMA request and the CPU must not clear this bit by writing 1 in BUF5I. 0 No occurrence of MB5 completing transmission/reception when MCR[RFEN]=0, or of frame(s) available in the 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. It generates a DMA request in case of MCR[RFEN] and MCR[DMA] are enabled.
4–1 BUF4TO1I	Buffer MB $_i$ Interrupt Or "reserved" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB1. **NOTE:** These flags are cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF4TO1I flags are reserved when CAN_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.
0 BUF0I	Buffer MB0 Interrupt Or Clear FIFO bit When the RFEN bit in MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB0. If the Rx FIFO is enabled, this bit is used to trigger the clear FIFO operation. This operation empties FIFO contents. Before performing this operation the CPU must service all FIFO related IFLAGs. When the bit MCR[DMA] is enabled this operation also clears the BUF5I flag and consequently abort the DMA request. The clear FIFO operation occurs when the CPU writes 1 in BUF0I. It is only allowed in Freeze Mode and is blocked by hardware in other conditions. 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.

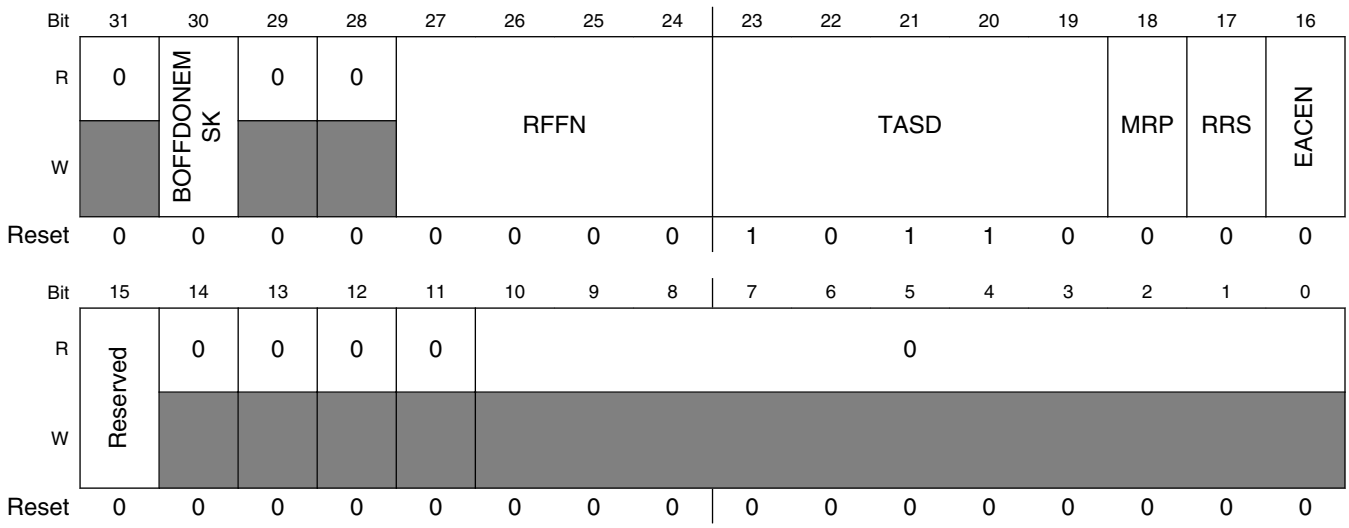
39.3.12 Control 2 register (CANx_CTRL2)

This register complements Control1 Register providing control bits for memory write access in Freeze Mode, for extending FIFO filter quantity, and for adjust the operation of internal FlexCAN processes like matching and arbitration.

The contents of this register are not affected by soft reset.

Memory map/register definition

Address: 4002_4000h base + 34h offset = 4002_4034h



CANx_CTRL2 field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 BOFFDONEMSK	Bus Off Done Interrupt Mask This bit provides a mask for the Bus Off Done Interrupt in CAN_ESR1 register. 0 Bus Off Done interrupt disabled. 1 Bus Off Done interrupt enabled.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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 chip. 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 CAN_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 corresponding to a number of filters not greater than the number of available memory words which can be calculated as follows: (SETUP_MB - 6) × 4 where SETUP_MB is the least between the parameter NUMBER_OF_MB and CAN_MCR[MAXMB]. The number of remaining Mailboxes available will be: (SETUP_MB - 8) - (RFFN × 2)

Table continues on the next page...

CANx_CTRL2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
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.

NOTE:
• The number of the last remaining available mailboxes is defined by the least value between the NUMBER_OF_MB minus 1 and the CAN_MCR[MAXMB] field.
• If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

<table>
<tr><th>RFFN[3:0]</th><th>Number of Rx FIFO filter elements</th><th>Message Buffers occupied by Rx FIFO and ID Filter Table</th><th>Remaining Available Mailboxes</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</th></tr>
<tr><td>0x0</td><td>8</td><td>MB 0-7</td><td>MB 8-15</td><td>Elements 0-7</td><td>none</td></tr>
<tr><td>0x1</td><td>16</td><td>MB 0-9</td><td>MB 10-15</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 12-15</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 14-15</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>none</td><td>Elements 0-15</td><td>Elements 16-39</td></tr>
</table>
</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. See Tx Arbitration start delay for more details. This field can be written only in Freeze mode because it is blocked by hardware in other modes.</td></tr>
<tr><td>18
MRP</td><td>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 be written only in Freeze mode because 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.</td></tr>
<tr><td>17
RRS</td><td>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 be written only in Freeze mode because it is blocked by hardware in other modes.

0 Remote Response Frame is generated.
1 Remote Request Frame is stored.</td></tr>
<tr><td>16
EACEN</td><td>Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes

This bit controls the comparison of IDE and RTR bits within 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 be written only in Freeze mode because it is blocked by hardware in other modes.</td></tr>
</table>

Table continues on the next page...

CANx_CTRL2 field descriptions (continued)

Field	Description
	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.
15 Reserved	This field is reserved. When writing to this field, always write the reset value.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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 0.
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.

39.3.13 Error and Status 2 register (CANx_ESR2)

This register reports some general status information.

Address: 4002_4000h base + 38h offset = 4002_4038h

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

CANx_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 CAN_ESR2[VPS] is asserted, this field indicates the lowest number inactive Mailbox (see the CAN_ESR2[IMB] bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CAN_CTRL1[LBUF] bit value. If CAN_CTRL1[LBUF] bit is negated then the Mailbox indicated is the one that has the greatest arbitration value (see the "Highest priority Mailbox first" section). If CAN_CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a

Table continues on the next page...

CANx_ESR2 field descriptions (continued)

Field	Description
	Tx Mailbox is being transmitted it is not considered in LPTM calculation. If CAN_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 CAN_ESR2[IMB] and CAN_ESR2[LPTM] contents are currently valid or not. It 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, that is, it is behind Tx Arbitration Pointer, during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox that is being transmitted then VPS is not asserted. This bit is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox. **NOTE:** CAN_ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of a MB that is blocked by abort mechanism. When CAN_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 CAN_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 CAN_ESR2[LPTM] is found and it is inactive. • If CAN_ESR2[IMB] is not asserted and a frame is transmitted successfully. This bit is cleared in all start of arbitration (see Section "Arbitration process"). **NOTE:** CAN_ESR2[LPTM] mechanism have the following behavior: if an MB is successfully transmitted and CAN_ESR2[IMB]=0 (no inactive Mailbox), then CAN_ESR2[VPS] and CAN_ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into CAN_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.

39.3.14 CRC Register (CANx_CRCR)

This register provides information about the CRC of transmitted messages. This register is updated at the same time the Tx Interrupt Flag is asserted.

Address: 4002_4000h base + 44h offset = 4002_4044h

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

CANx_CRCR 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 MBCRC	CRC Mailbox This field indicates the number of the Mailbox corresponding to the value in CAN_CRCR[TXCRC] field.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXCRC	Transmitted CRC value This field indicates the CRC value of the last transmitted message.

39.3.15 Rx FIFO Global Mask register (CANx_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 CAN_CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 4002_4000h base + 48h offset = 4002_4048h

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	FGM																															
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.

CANx_RXFGMASK field descriptions

Field	Description
FGM	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 (CAN_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], FGM[15:8], FGM[7:0]	

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

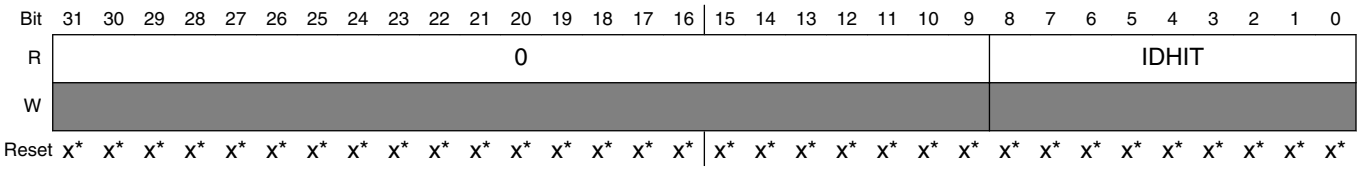
1. If CAN_MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If CAN_MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

39.3.16 Rx FIFO Information Register (CANx_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: 4002_4000h base + 4Ch offset = 4002_404Ch



* Notes:
- x = Undefined at reset.

CANx_RXFIR field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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 CAN_IFLAG1[BUF5I] is asserted.

39.3.17 CAN Bit Timing Register (CANx_CBT)

This register is an alternative way to store the CAN bit timing variables described in CAN_CTRL1 register. EPRESDIV, EPROPSEG, EPSEG1, EPSEG2 and ERJW are extended versions of PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW bit fields respectively.

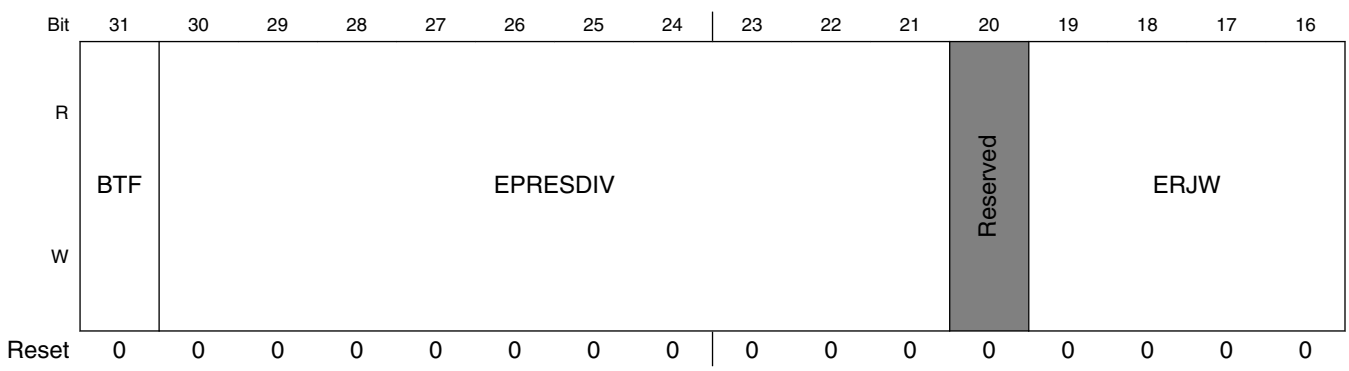
The BTF bit selects the use of the timing variables defined in this register.

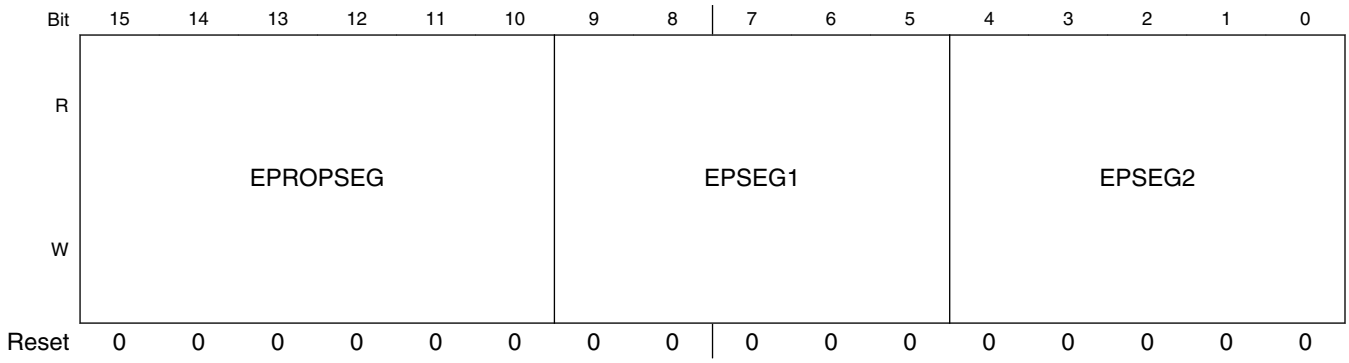
The contents of this register are not affected by soft reset.

NOTE

The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.

Address: 4002_4000h base + 50h offset = 4002_4050h





CANx_CBT field descriptions

Field	Description
31 BTF	Bit Timing Format Enable Enables the use of extended CAN bit timing fields EPRESDIV, EPROPSEG, EPSEG1, EPSEG2 and ERJW replacing the CAN bit timing variables defined in CAN_CTRL1 register. This field can be written in Freeze mode only. 0 Extended bit time definitions disabled. 1 Extended bit time definitions enabled.
30–21 EPRESDIV	Extended Prescaler Division Factor This 10-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PRESDIV] value range. 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 (see Protocol timing). This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (EPRESDIV + 1)
20 Reserved	This field is reserved.
19–16 ERJW	Extended Resync Jump Width This 4-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[RJW] value range. One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = ERJW + 1.
15–10 EPROPSEG	Extended Propagation Segment This 6-bit field defines the length of the Propagation Segment in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PROPSEG] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (EPROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.
9–5 EPSEG1	Extended Phase Segment 1

Table continues on the next page...

CANx_CBT field descriptions (continued)

Field	Description
	This 5-bit field defines the length of Phase Segment 1 in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PSEG1] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG1 + 1) × Time-Quanta. Time-Quantum = one Sclock period.
EPSEG2	Extended Phase Segment 2 This 5-bit field defines the length of Phase Segment 2 in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PSEG2] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG2 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

39.3.18 Rx Individual Mask Registers (CANx_RXIMR*n*)

The RX Individual Mask Registers are used to store the acceptance masks for ID filtering in Rx MBs and the Rx FIFO.

When the Rx FIFO is disabled (CAN_MCR[RFEN] bit is negated), an individual mask is provided for each available Rx Mailbox on a one-to-one correspondence. When the Rx FIFO is enabled (CAN_MCR[RFEN] bit is asserted), an individual mask is provided for each Rx FIFO ID Filter Table Element on a one-to-one correspondence depending on the setting of CAN_CTRL2[RFFN] (see Rx FIFO).

CAN_RXIMR0 stores the individual mask associated to either MB0 or ID Filter Table Element 0, CAN_RXIMR1 stores the individual mask associated to either MB1 or ID Filter Table Element 1 and so on.

CAN_RXIMR registers can only be accessed by the CPU while the module is in Freeze mode, otherwise, they are blocked by hardware. These registers are not affected by reset. They are located in RAM and must be explicitly initialized prior to any reception.

It is possible for the RXIMR memory region to be accessed as general purpose memory. See Bus interface for more information.

Address: 4002_4000h base + 880h offset + (4d × 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																															
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.

CANx_RXIMR*n* field descriptions

Field	Description
MI	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.

39.3.36 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 0x80 to 0x17F is used by the mailboxes.

Table 39-3. Message buffer structure

	31	30	29	28	27 … 24	23	22	21	20	19 … 18	17 … 16	15 … 8	7 … 0
0x0	EDL	BRS	ESI		CODE		SRR	IDE	RTR	DLC		TIME STAMP	
0x4	PRIO			ID (Standard/Extended)							ID (Extended)		
0x8	Data Byte 0					Data Byte 1						Data Byte 2	Data Byte 3
0xC	Data Byte 4					Data Byte 5						Data Byte 6	Data Byte 7

= Unimplemented or Reserved

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 39-4 and Table 39-5. See Functional description for additional information.

Table 39-4. 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.
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.

Table continues on the next page...

Table 39-4. 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
		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.
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 CAN_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]=1: 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, see "Control 2 Register (CAN_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 39-5. 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.
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 39-4, Table 39-5, and the description of the RRS bit in Control 2 Register (CAN_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 0x8 through 0xF of the MB space (see Table 39-3) . 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 39-6).

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 CAN_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 to 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.

Table 39-6. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0 to 1
3	DATA BYTE 0 to 2
4	DATA BYTE 0 to 3
5	DATA BYTE 0 to 4
6	DATA BYTE 0 to 5
7	DATA BYTE 0 to 6
8 or above	DATA BYTE 0 to 7

39.3.37 Rx FIFO structure

When the CAN_MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the reception FIFO engine.

The region 0x80-0x8C 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 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x17C (normally occupied by MBs 6–15) depending on the CAN_CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 40 table elements) that specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, 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 39-7. Rx FIFO structure

	31–28	24–23	22	21	20	19–18	17–16	15–8	7–0
0x80	IDHIT		SRR	IDE	RTR	DLC		TIME STAMP	
0x84		ID standard						ID extended	
0x88	Data byte 0		Data byte 1					Data byte 2	Data byte 3
0x8C	Data byte 4		Data byte 5					Data byte 6	Data byte 7
0x90 to 0xDC	Reserved								
0xE0	ID filter table element 0								
0xE4	ID filter table element 1								
0xE8 to 0x2D4	ID filter table elements 2 to 125								
0x2D8	ID filter table element 126								
0x2DC	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 CAN_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 39-8. 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		RTR	IDE	RXIDB_1		

Table continues on the next page...

Table 39-8. ID table structure (continued)

C			(standard = 29–19, extended = 29–16)			(standard = 13–3, extended = 13–0)
	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.

IDHIT — Identifier Acceptance Filter Hit Indicator

This 9-bit field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. See Rx FIFO for more information.

39.4 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 39-4). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 39-5).

39.4.1 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 bit to be asserted by polling the CAN_IFLAG register or by the interrupt request if enabled by the respective IMASK bit. Then read back the CODE field to check if the transmission was aborted or transmitted (see Transmission abort mechanism). If backwards compatibility is

desired (CAN_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 participates in the arbitration process and is eventually 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 39-4 and Table 39-5 in Message buffer structure).

When the Abort feature is enabled (CAN_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 bit before starting to prepare this MB for a new transmission or reception.

39.4.2 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 CAN_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 CAN_CTRL1[LBUF] and CAN_MCR[LPRIOEN] bits settings.

39.4.2.1 Lowest-number Mailbox first

If CAN_CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. CAN_MCR[LPRIOEN] bit has no effect when CAN_CTRL1[LBUF] is asserted.

39.4.2.2 Highest-priority Mailbox first

If CAN_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 CAN_MCR[LPRIOEN] bit setting.

39.4.2.2.1 Local Priority disabled

If CAN_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 39-9. 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)

39.4.2.2.2 Local Priority enabled

If Local Priority is desired CAN_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 39-10. 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.

39.4.2.3 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 C/S word of the corresponding MB is blocked (if the AEN bit in CAN_MCR register is asserted). Write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the CPU
- 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.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. CAN_CTRL2[TASD] bit value may be changed to optimize the arbitration start point.
- During CAN BusOff state from TX_ERR_CNT=124 to 128. CAN_CTRL2[TASD] bit 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 the arbitration process has finished, then the 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 exiting Freeze mode (first bit of the WaitForBusIdle state). If there is a re-synchronization during WaitForBusIdle, the arbitration process is restarted.

Arbitration process stops in the following situations:

- 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.

39.4.3 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 for Classical CAN message format) 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 39-4 and Table 39-5 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 poll for 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 39-4 . 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. When CAN_MCR[SRXDIS] bit is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching Rx Mailbox, and no interrupt flag or interrupt signal will be generated. Otherwise, when CAN_MCR[SRXDIS] bit is deasserted, FlexCAN can receive frames transmitted by itself if there exists a matching Rx Mailbox.

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 BUF5I bit "Frames available in Rx FIFO" bit in the CAN_IFLAG1 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 CAN_RXFIR register (optional)
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to CAN_IFLAG1[BUF5I] bit (mandatory: releases the MB and allows the CPU to read the next Rx FIFO entry)

When CAN_MCR[DMA] is asserted, upon receiving a frame in FIFO, CAN_IFLAG1[BUF5I] generates a DMA request and does not generate a CPU interrupt (see Rx FIFO under DMA Operation). The CAN_IMASK1 bits in Rx FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1. Read the Control and Status word (read 0x80 address, optional)
2. Read the ID field (read 0x84 address, optional)
3. Read all Data Bytes (start read at 0x88 address, optional)
4. Read the last Data Bytes (read 0x8C address is mandatory)

39.4.4 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. 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 always looks 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 Rx SMB are 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 are 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 the 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 Rx 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 39-11. 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
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 Rx SMB contents with the MB contents regardless the masks.
3. no_cmp: The Rx SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the Rx 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 CAN_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 CAN_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 of 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 CAN_MCR[IRMQ] bit:
- - If CAN_MCR[IRMQ] bit is negated, the matching winner is the first matched Mailbox
  - If CAN_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 39-12. 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)
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 (CAN_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 finds 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 finds MB number 2 again, but it is not "free-to-receive", so it keeps looking, finds MB number 5 and stores 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 are 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. See the description of the Rx Individual Mask Registers (CAN_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 a "don't care". 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 (CAN_RXFGMASK, CAN_RXMGMASK, CAN_RX14MASK and CAN_RX15MASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the CAN_MCR Register is negated.

39.4.5 Move process

There are two types of move process: move-in and move-out.

39.4.5.1 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 CAN_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 (CAN_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. Push IDHIT into the RXFIR FIFO if the message is destined to the Rx FIFO.
2. Read DATA0-3 and DATA4-7 words from the Rx SMB.
3. Write DATA0-3 and DATA4-7 words to the Rx Mailbox
4. Read the Control/Status and ID words from the Rx SMB.
5. Write Control/Status and ID words to the Rx Mailbox, and update 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.

39.4.5.2 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.

39.4.6 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.

39.4.6.1 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:

- CAN_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.

Active MBs configured for transmission must be aborted first before they can 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 Tx SMB for transmission, the write operation is blocked and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze mode
- The module enters the 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.

39.4.6.2 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 instructions 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 have the following consequences:

- 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 setting the respective IFLAG

In order to perform a *safe inactivation* and avoid the above consequences for Tx Mailboxes, the CPU must use the Transmission Abort mechanism (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.

39.4.6.3 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 the 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 prevents a new frame from being written into the MB while the CPU is reading it.

NOTE

The locking mechanism applies only to Rx MBs that are not part of the 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 Rx 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 Rx 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 Rx 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.

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.

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 Rx 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 (CAN_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 takes place only when the module resumes to Normal or Freeze modes.

39.4.7 Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the CAN_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 CAN_IFLAG1[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 CAN_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 CAN_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 CAN_IFLAG1[BUF5I] is asserted.

The CAN_IFLAG1[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 CAN_IFLAG1[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 CAN_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 read in the IDHIT field from C/S word, as shown in the Rx FIFO Structure description. Another way the CPU can obtain this information is by accessing the CAN_RXFIR register. The CAN_RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid while the CAN_IFLAG1[BUF5I] flag is asserted. The CAN_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 16 elements of the filter table are individually affected by the Individual Mask Registers (CAN_RXIMRx), according to the setting of CAN_CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If the CAN_MCR[IRMQ] bit is negated, then the FIFO filter table is affected by CAN_RXFGMASK.

39.4.7.1 Rx FIFO under DMA Operation

The receive-only FIFO can support DMA, this feature is enabled by asserting both the CAN_MCR[RFEN] and CAN_MCR[DMA] bits. The reset value of CAN_MCR[DMA] bit is zero to maintain backward compatibility with previous versions of the module that did not have the DMA feature.

The DMA controller can read the received message by reading a Message Buffer structure at the FIFO output port at the 0x80-0x8C address range.

When CAN_MCR[DMA] is asserted the CPU must not access the FIFO output port address range. Before enabling the CAN_MCR[DMA], the CPU must service the IFLAGs asserted in the Rx FIFO region. Otherwise, these IFLAGs may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before disabling the CAN_MCR[DMA], the CPU must perform a clear FIFO operation.

The CAN_IFLAG1[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO, consequently a DMA request is generated simultaneously. Upon receiving the request, the DMA controller can read the message (accessing the output of the FIFO as a Message Buffer). The DMA reading process must end by reading address 0x8C, which clears the CAN_IFLAG1[BUF5I] and updates both the FIFO output with the next message (if FIFO is not empty) and the CAN_RXFIR register with the attributes of the new message. If there are more messages stored in the FIFO, the CAN_IFLAG1[BUF5I] will be re-asserted and another DMA request is issued. Otherwise, the flag remains negated.

NOTE

CAN_RXFIR register contents cannot be read after DMA completes the FIFO read. The IDHIT information is also available in the C/S word at address 0x080 (see Rx FIFO structure.

The CAN_IFLAG1[BUF6I] and CAN_IFLAG1[BUF7I] are not used when the DMA feature is enabled.

When FlexCAN is working with DMA, the CPU does not receive any Rx FIFO interruption and must not clear the related IFLAGs. In addition, the related IMASKs are not used to mask the generation of DMA requests.

39.4.7.2 Clear FIFO Operation

When CAN_MCR[RFEN] is asserted, the clear FIFO operation is a feature used to empty FIFO contents. With CAN_MCR[RFEN] asserted the Clear FIFO occurs when the CPU writes 1 in CAN_IFLAG1[BUF0I]. This operation can only be performed in Freeze

Mode and is blocked by hardware in other modes. This operation does not clear the FIFO IFLAGs, consequently the CPU must service all FIFO IFLAGs before execute the clear FIFO task.

When Rx FIFO is working with DMA, the clear FIFO operation clears the CAN_IFLAG1[BUF5I] and the DMA request is canceled.

CAUTION

Clear FIFO operation does not clear IFLAGs, except when CAN_MCR[DMA] is asserted, in this case only the CAN_IFLAG1[BUF5I] is cleared.

39.4.8 CAN protocol related features

This section describes the CAN protocol related features.

39.4.8.1 Remote frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by configuring 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 is received and matches a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as a 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.

39.4.8.2 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

39.4.8.3 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 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 Free Running Timer is not incremented during Disable, Doze, Stop, and Freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in Control 1 Register (CAN_CTRL1).

39.4.8.4 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 CAN_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 (MDIS bit set in the Module Configuration Register).

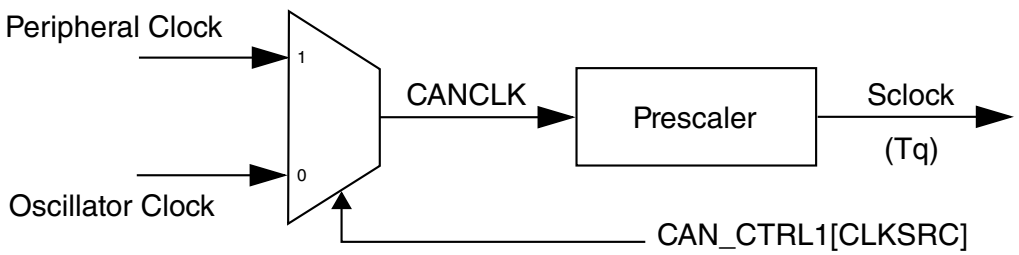


Figure 39-2. CAN engine clocking scheme

The oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than the peripheral clock.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control 1 Register (CAN_CTRL1) has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW.

The CAN Bit Timing register (CAN_CBT) extends the range of the CAN bit timing variables in CAN_CTRL1.

The PRESDIV field (as well as its extended range EPRESDIV) defines the Prescaler Value (see the equation below) that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time handled by the CAN engine.

$$Tq = \frac{(\text{PRESDIV} + 1)}{f_{\text{CANCLK}}}$$

The bit rate, which defines the rate the CAN message is either received or transmitted, is given by the formula:

CAN Bit Time = (Number of Time Quanta in 1 bit time) * Tq

$$\text{Bit Rate} = \frac{1}{\text{CAN Bit Time}}$$

A bit time is subdivided into three segments[1] (see Figure 39-3 and Table 39-13):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the 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 CAN_CTRL1 Register so that their sum (plus 2) is in the range of 2 to 16 time quanta. When CAN_CBT[BTF] bit is asserted, FlexCAN uses EPROPSEG and EPSEG1 fields from CAN_CBT register so that their sum (plus 2) is in the range of 2 to 96 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 CAN_CTRL1 Register (plus 1) to be 2 to 8 time quanta long. When CAN_CBT[BTF] bit is asserted, FlexCAN uses EPSEG2 fields of CAN_CBT register so that its value (plus 1) is in the range of 2 to 32 time quanta. The Time Segment 2 cannot be smaller than the Information Processing Time (IPT), which value is 2 time quanta in FlexCAN.

NOTE

The bit time defined by the above time segments must not be smaller than 5 time quanta. For bit time calculations, use an Information Processing Time (IPT) of 2, which is the value implemented in the FlexCAN module.

1. For further explanation of the underlying concepts, see ISO 11898-1. See also the CAN 2.0A/B protocol specification for bit timing.

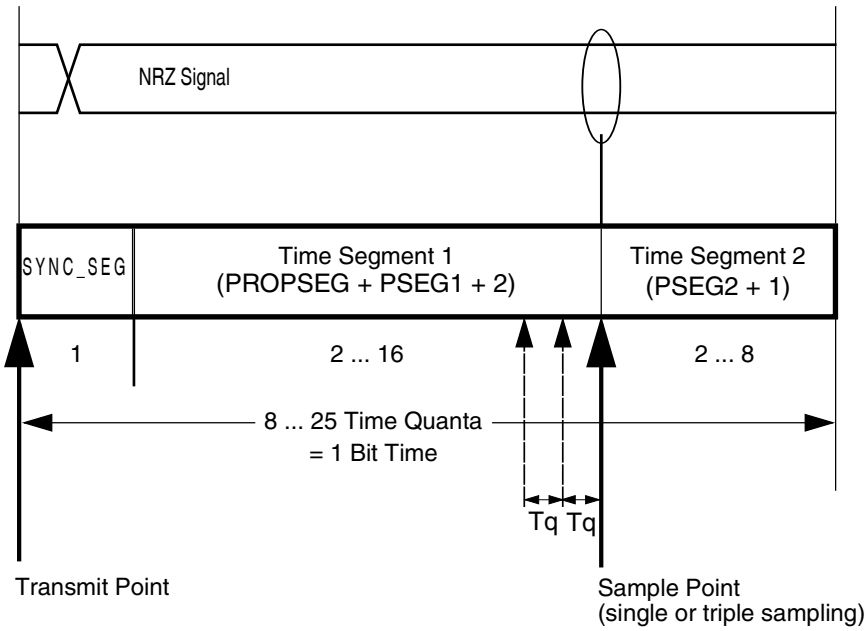


Figure 39-3. Segments within the bit time (example using CAN_CTRL1 bit timing variables for Classical CAN format)

Table 39-13. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
TSEG1	Corresponds to the sum of PROPSEG and PSEG1.
TSEG2	Corresponds to the PSEG2 value.
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 some examples of the CAN compliant segment settings for Classical CAN format (Bosch CAN 2.0B) messages.

Table 39-14. 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).

Whenever CAN bit is used as a measure of time duration (e.g. estimating the occurrence of a CAN bit event in a message), the number of peripheral clocks in one CAN bit (NumClkBit) can be calculated as:

$$\text{NumClkBit} = \frac{f_{SYS}}{f_{CANCLK}} \text{ X } (\text{PRESDIV} + 1) \text{ X } (\text{PROPSEG} + \text{PSEG1} + \text{PSEG2} + 4)$$

where:

- NumClkBit 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 CAN_CTRL1[PSEG1] field;
- PSEG2 is the value in CAN_CTRL1[PSEG2] field;
- PROPSEG is the value in CAN_CTRL1[PROPSEG] field;
- PRESDIV is the value in CAN_CTRL1[PRESDIV] field.

The formula above is also applicable to the alternative CAN bit timing variables described in the CAN Bit Timing Register (CAN_CBT).

For example, 180 CAN bits = (180 x NumClkBit) peripheral clock periods.

39.4.8.5 Arbitration and matching timing

During normal reception and transmission, 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.

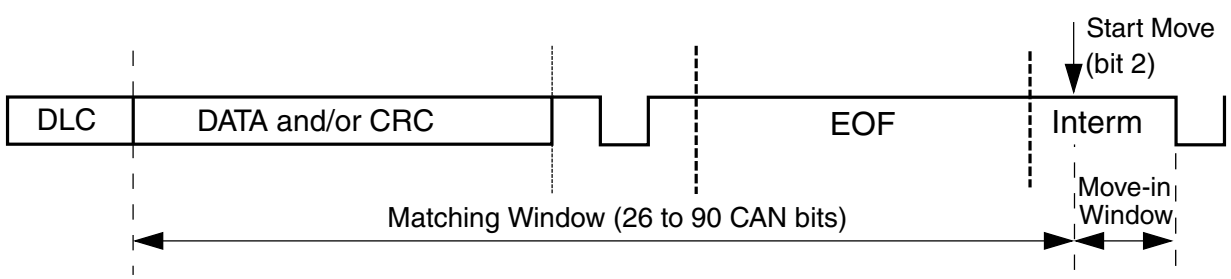


Figure 39-4. Matching and move-in time windows

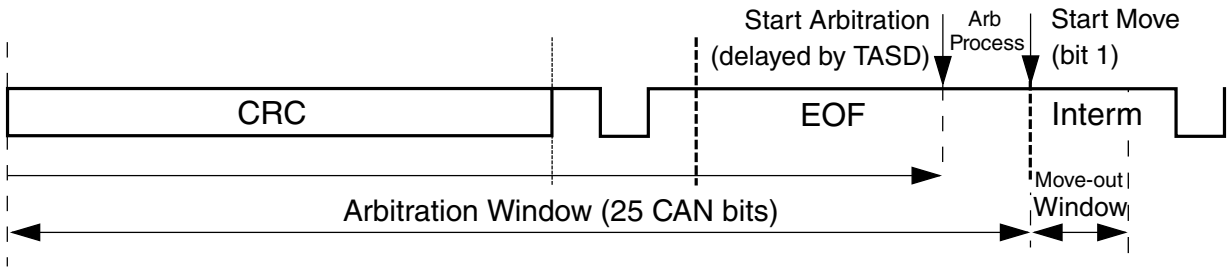


Figure 39-5. Arbitration and move-out time windows

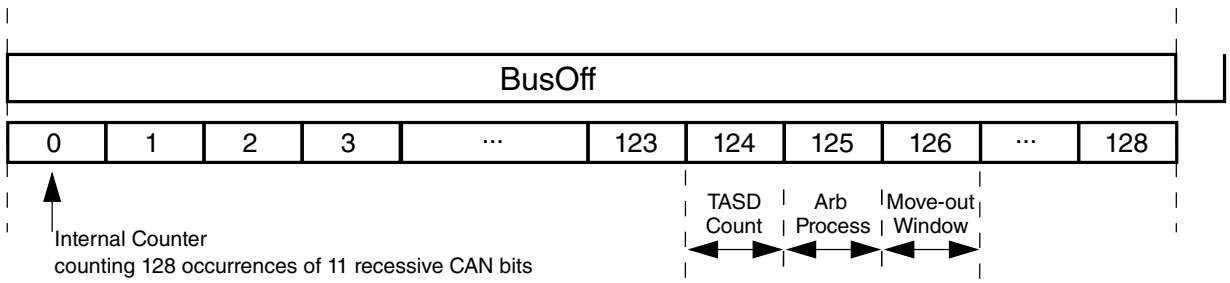


Figure 39-6. Arbitration at the end of bus off and move-out time windows

NOTE

In the preceding figures, the matching and arbitration timing does not take into account the delay caused by the concurrent memory access due to the CPU or other internal FlexCAN sub-blocks.

39.4.8.6 Tx Arbitration start delay

The Tx Arbitration Start Delay (TASD) bit field in Control 2 register (CAN_CTRL2[TASD]) is a variable that indicates the number of CAN bits used by FlexCAN to delay the Tx Arbitration process start point from the first bit of CRC field of the current frame. This variable can be written only in Freeze mode because it is blocked by hardware in other modes.

The transmission performance is impacted by the ability of the CPU to reconfigure Message Buffers (MBs) for transmission after the end of the internal Arbitration process, where FlexCAN finds the winner MB for transmission (see Arbitration process). If the Arbitration ends too early before the first bit of Intermission field, then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is no longer the best candidate to be transmitted.

TASD is useful to optimize the transmission performance by defining the Arbitration start point, as shown in the next figure, based on factors such as:

- The peripheral-to-oscillator clock ratio

- CAN bit timing variables that determine the CAN bit rate
- The number of Message Buffers (MBs) in use by the Matching and Arbitration processes.

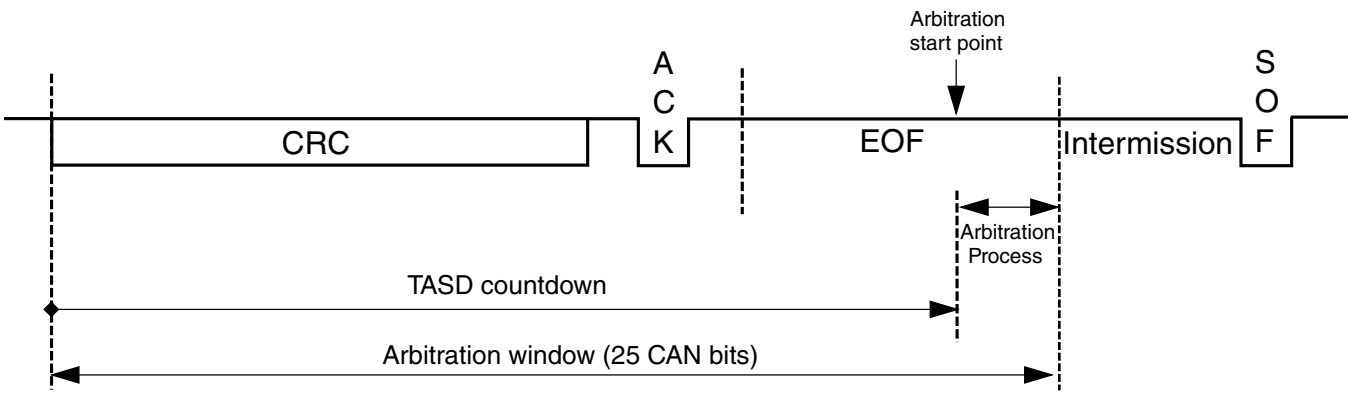


Figure 39-7. Optimal Tx Arbitration start point

The duration of an Arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and to the CAN bit 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. For instance, if there are few MBs and the peripheral/oscillator clock ratio is high and the CAN baud rate is low, then the Arbitration can be placed closer to the frame's end, adding more delay to its start point, and vice-versa.

If TASD is set to 0 then the Arbitration start is not delayed and more time is reserved for Arbitration. On the other hand, if TASD is close to 24 then the CPU can configure a Tx MB later and less time is reserved for Arbitration. If too little time is reserved for Arbitration the FlexCAN may be not be able to find a winner MB in time to be transmitted with the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal TASD value can be calculated as follows:

$$TASD = 25 - \left(\frac{f_{CANCLK} \times [\, MAXMB + 3 - (RFEN \times 8) - (RFEN \times RFFN \times 2)\,] \times 2}{f_{SYS} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)\,] \times (PRESDIV + 1)} \right)$$

where:

- MAXMB is the value in CAN_CTRL1[MAXMB] field
- f_{CANCLK} is the oscillator clock, in Hz

- f_{SYS} is the peripheral clock, in Hz
- RFEN is the value in CAN_CTRL1[RFEN] bit
- RFFN is the value in CAN_CTRL2[RFFN] field
- PSEG1 is the value in CAN_CTRL1[PSEG1] field
- PSEG2 is the value in CAN_CTRL1[PSEG2] field
- PROPSEG is the value in CAN_CTRL1[PROPSEG] field
- PRESDIV is the value in CAN_CTRL1[PRESDIV] field

See also Protocol timing for more details.

39.4.9 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 CAN_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 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.

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the time slot of one CAN frame, comprised of a number of CAN bits. In order to have sufficient time to do that, the following requirements must be observed:

- The peripheral clock frequency can not be smaller than the oscillator clock frequency
- For 16 Mailboxes, the minimum number of peripheral clocks per CAN bit is 16

The minimum number of peripheral clocks per CAN bit determines the minimum peripheral clock frequency for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit, that can be defined by adjusting one or more of the bit timing values contained in either the Control 1 Register (CAN_CTRL1) or CAN Bit Time register (CAN_CBT). The time quantum (Tq) is defined in Protocol timing. 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.

39.4.10 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.

39.4.10.1 Freeze mode

This mode is requested either by the CPU through the assertion of the HALT bit in the CAN_MCR Register or when the chip is put into Debug mode. In both cases it is also necessary that the FRZ bit is asserted in the CAN_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, 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 does not prevent going to Freeze mode.
- 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 CAN_MCR

After requesting Freeze mode, the user must wait for the FRZ_ACK bit to be asserted in CAN_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 CAN_CTRL1[CLKSRC] 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 CAN_MCR Register
- The chip 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.

39.4.10.2 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 CAN_MCR[MDIS] bit, and the acknowledgement is obtained through the assertion by the FlexCAN of the CAN_MCR[LPMACK] bit. 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 LPMACK bit and negates the FRZACK 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 CAN_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 LPMACK bit after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

39.4.10.3 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 CAN_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 CAN_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 CAN_MCR Register
- Self Wake mechanism

In the Self Wake mechanism, if the SLFWAK bit in CAN_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 CAN_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 39-15. 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 (CAN_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.

39.4.10.4 Stop mode

This is a system low-power mode in which all chip 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 CAN_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 CAN_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 CAN_ESR Register and, if enabled by the WAKMSK bit in CAN_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 39-16. Wake-up from Stop Mode

SLFWAK	WAKINT	WAKMSK	Chip clocks enabled	Wake-up interrupt generated
0	-	-	No	No
0	-	-	No	No
1	0	0	No	No

Table continues on the next page...

Table 39-16. Wake-up from Stop Mode (continued)

SLFWAK	WAKINT	WAKMSK	Chip clocks enabled	Wake-up interrupt generated
1	0	1	No	No
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 (CAN_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.

39.4.11 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Bus Off Done, 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 an assigned flag bit in the CAN_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 (CAN_MCR[RFEN] = 1) and DMA is disabled (CAN_MCR[DMA] = 0), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 7 of the CAN_IFLAG1 register 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 (CAN_IFLAG1) for more information.

If both Rx FIFO and DMA are enabled (CAN_MCR[RFEN] and CAN_MCR[DMA] = 1) the FlexCAN does not generate any FIFO interrupt. Bit 5 of the CAN_IFLAG1 register still indicates "Frames Available in FIFO" and generates a DMA request. Bits 7, 6, 4-0 are unused.

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 CAN_IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Bus Off Done, Error, Wake Up, Tx Warning and Rx Warning generate interrupts like the MB interrupt sources, and can be read from CAN_ESR1 register. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the CAN_CTRL1 Register; the Wake-Up interrupt mask bit is located in the CAN_MCR.

39.4.12 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.
- It is possible for the RXIMR memory region to be considered as general purpose memory and available for access. There are two ways of doing this:
  a. If CAN_MCR[IRMQ] is cleared, the individual masks (RXIMR) are disabled. In this case the RXIMR memory region is considered as general purpose memory.
  b. If CAN_MCR[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's RAM can support up to 16 MBs, CAN_CTRL2[RFFN] is 0x0, and CAN_MCR[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.

39.5 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

39.5.1 FlexCAN initialization sequence

The FlexCAN module may be reset in three ways:

- Chip 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 39-2 to see what registers are affected by soft reset.
- Chip 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 CAN_MCR[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 should be selected while the module is in Disable mode (see CAN_CTRL1[CLKSRC] bit). After the clock source is selected and the module is enabled (CAN_MCR[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 CAN_MCR Register are set, the internal state machines are disabled and the FRZACK and NOTRDY bits in the CAN_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 (CAN_MCR)
  - 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
  - If Rx FIFO is enabled and DMA is required, set DMA bit
  - Enable the abort mechanism by setting the AEN bit
  - Enable the local priority feature by setting the LPRIOEN bit
- Initialize the Control 1 Register (CAN_CTRL1) and optionally the CAN Bit Timing Register (CAN_CBT).
  - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
  - Optionally determine the bit timing parameters: EPROPSEG, EPSEG1, EPSEG2, ERJW
  - Determine the bit rate by programming the PRESDIV field and optionally the EPRESDIV 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 (CAN_RXIMRn)
- Set required interrupt mask bits in the CAN_IMASK Registers (for all MB interrupts), in CAN_MCR Register for Wake-Up interrupt and in CAN_CTRL1 / CAN_CTRL2 Registers (for Bus Off and Error interrupts)
- Negate the HALT bit in CAN_MCR

After the last step listed above, FlexCAN attempts to synchronize to the CAN bus.

Chapter 40
Serial Peripheral Interface (SPI)

40.1 Introduction

NOTE

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

The serial peripheral interface (SPI) module provides a synchronous serial bus for communication between a chip and an external peripheral device.

40.1.1 Block Diagram

The block diagram of this module is as follows:

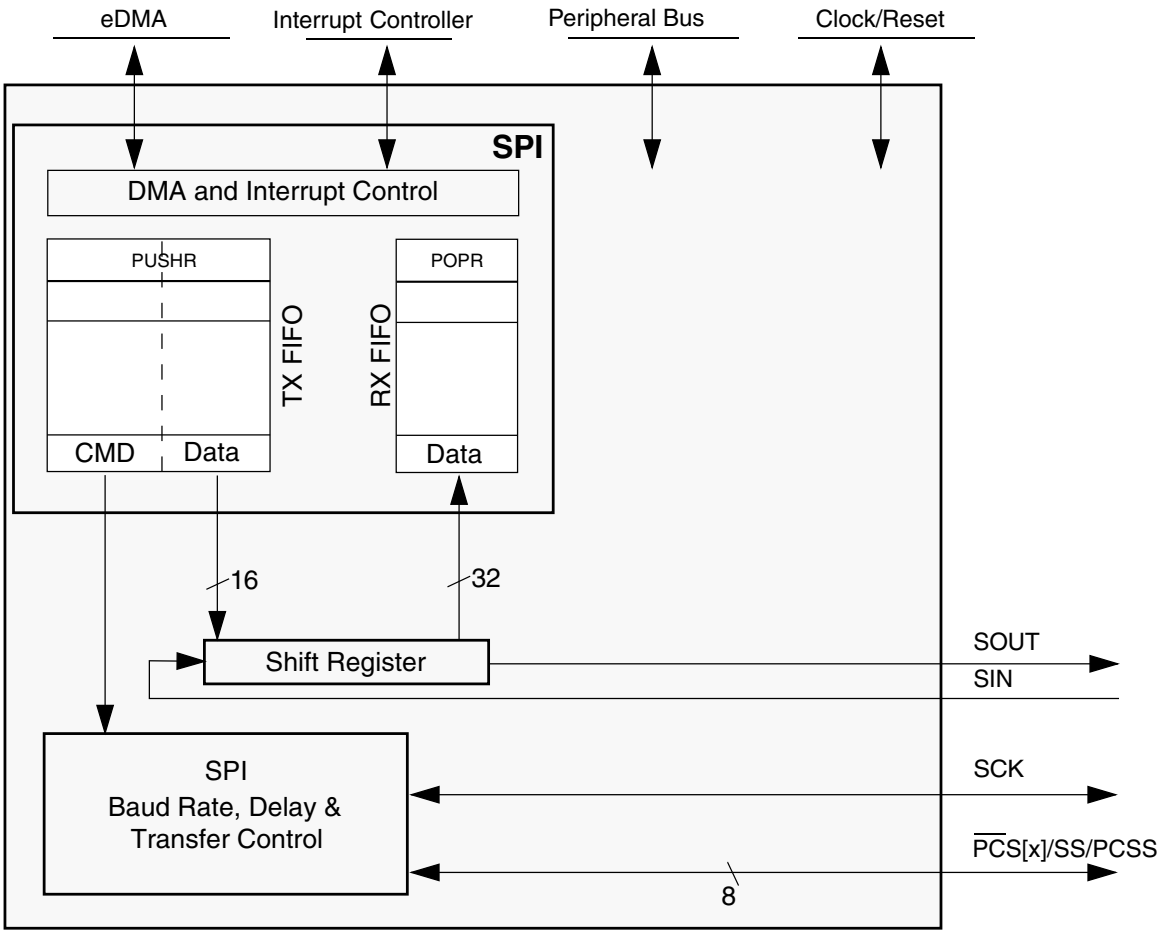


Figure 40-1. SPI Block Diagram

40.1.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master mode
- Slave mode
- Data streaming operation in Slave mode with continuous slave selection
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of 4 entries
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 4 entries
- Asynchronous clocking scheme for Register and Protocol Interfaces

- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
  - two transfer attribute registers
  - Serial clock (SCK) with programmable polarity and phase
  - Various programmable delays
  - Programmable serial frame size: 4 to 16
    - SPI frames longer than 16 bits can be supported using the continuous selection format.
  - Continuously held chip select capability
- 5 peripheral chip selects (PCSes), expandable to 32 with external demultiplexer
- Deglitching support for up to 16 peripheral chip selects (PCSes) with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
- Interrupt conditions:
  - End of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - Transfer of current frame complete (TCF)
  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - Frame received while Receive FIFO is full (RFOF)
- Global interrupt request line
- Modified SPI transfer formats for communication with slower peripheral devices
- Power-saving architectural features:

- Support for Stop mode
- Support for Doze mode

40.1.3 Interface configurations

40.1.3.1 SPI configuration

The Serial Peripheral Interface (SPI) configuration allows the module to send and receive serial data. This configuration allows the module to operate as a basic SPI block with internal FIFOs supporting external queue operation. Transmitted data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the Receive FIFO and write transmit data to the Transmit FIFO.

For queued operations, the SPI queues can reside in system RAM, external to the module. Data transfers between the queues and the module FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, SPI, and external queues in system RAM.

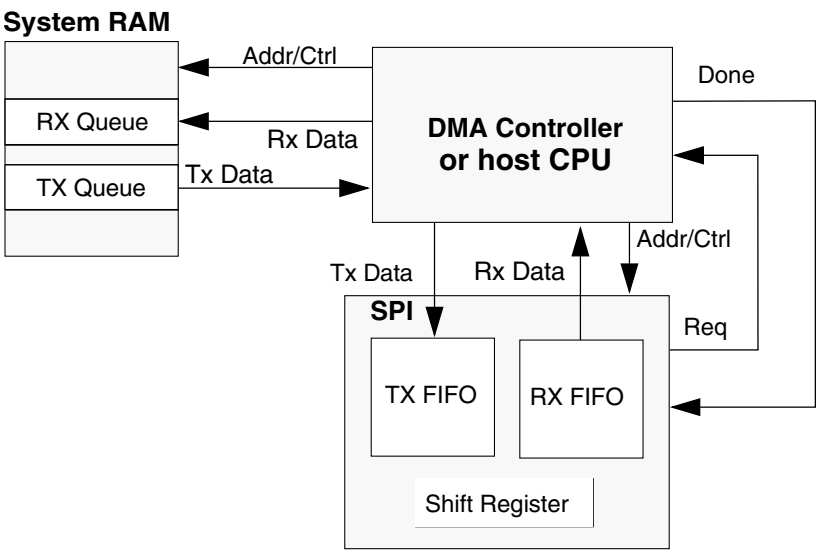


Figure 40-2. SPI with queues and DMA

40.1.4 Modes of Operation

The module supports the following modes of operation that can be divided into two categories:

- Module-specific modes:

    - Master mode
    - Slave mode
    - Module Disable mode
- Chip-specific modes:
    - External Stop mode
    - Debug mode

The module enters module-specific modes when the host writes a module register. The chip-specific modes are controlled by signals external to the module. The chip-specific modes are modes that a chip may enter in parallel to the block-specific modes.

40.1.4.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, these signals are controlled by the module and configured as outputs:

- SCK
- SOUT
- PCS[*x*]

40.1.4.2 Slave Mode

Slave mode allows the module to communicate with SPI bus masters. In this mode, the module responds to externally controlled serial transfers. The SCK signal and the PCS[0]/$\overline{\text{SS}}$ signals are configured as inputs and driven by an SPI bus master.

40.1.4.3 Module Disable Mode

The Module Disable mode can be used for chip power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

40.1.4.4 External Stop Mode

External Stop mode is used for chip power management. The module supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, it acknowledges the request and completes the transfer that is in progress. When the module reaches the frame boundary, it signals that the protocol clock to the module may be shut off.

40.1.4.5 Debug Mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls module behavior in the Debug mode:

- If the bit is set, the module stops all serial transfers, when the chip is in debug mode.
- If the bit is cleared, the chip debug mode has no effect on the module.

40.2 Module signal descriptions

This table describes the signals on the boundary of the module that may connect off chip (in alphabetical order).

Table 40-1. Module signal descriptions

Signal	Master mode	Slave mode	I/O
PCS0/$\overline{SS}$	Peripheral Chip Select 0 (O)	Slave Select (I)	I/O
PCS[1:3]	Peripheral Chip Selects 1–3	(Unused)	O
SCK	Serial Clock (O)	Serial Clock (I)	I/O
SIN	Serial Data In	Serial Data In	I
SOUT	Serial Data Out	Serial Data Out	O

40.2.1 PCS0/$\overline{SS}$—Peripheral Chip Select/Slave Select

Master mode: Peripheral Chip Select 0 (O)—Selects an SPI slave to receive data transmitted from the module.

Slave mode: Slave Select (I)—Selects the module to receive data transmitted from an SPI master.

NOTE

Do not tie the SPI slave select pin to ground. Otherwise, SPI cannot function properly.

40.2.2 PCS1–PCS3—Peripheral Chip Selects 1–3

Master mode: Peripheral Chip Selects 1–3 (O)—Select an SPI slave to receive data transmitted by the module.

Slave mode: Unused

40.2.3 SCK—Serial Clock

Master mode: Serial Clock (O)—Supplies a clock signal from the module to SPI slaves.

Slave mode: Serial Clock (I)—Supplies a clock signal to the module from an SPI master.

40.2.4 SIN—Serial Input

Master mode: Serial Input (I)—Receives serial data.

Slave mode: Serial Input (I)—Receives serial data.

40.2.5 SOUT—Serial Output

Master mode: Serial Output (O)—Transmits serial data.

Slave mode: Serial Output (O)—Transmits serial data.

40.3 Memory Map/Register Definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Any Write access to the POPR and RXFRn also results in a transfer error.

SPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_C000	Module Configuration Register (SPI0_MCR)	32	R/W	0000_4001h	40.3.1/1003
4002_C008	Transfer Count Register (SPI0_TCR)	32	R/W	0000_0000h	40.3.2/1007
4002_C00C	Clock and Transfer Attributes Register (In Master Mode) (SPI0_CTAR0)	32	R/W	7800_0000h	40.3.3/1007

Table continues on the next page...

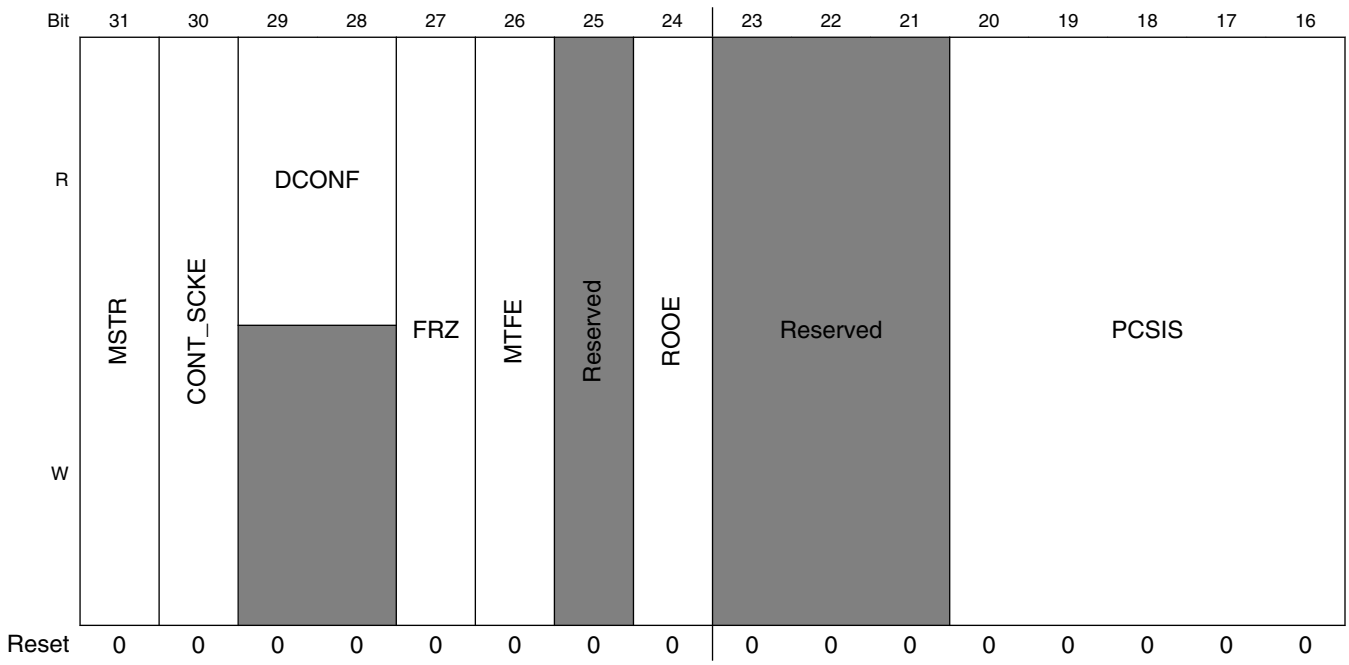
SPI memory map (continued)

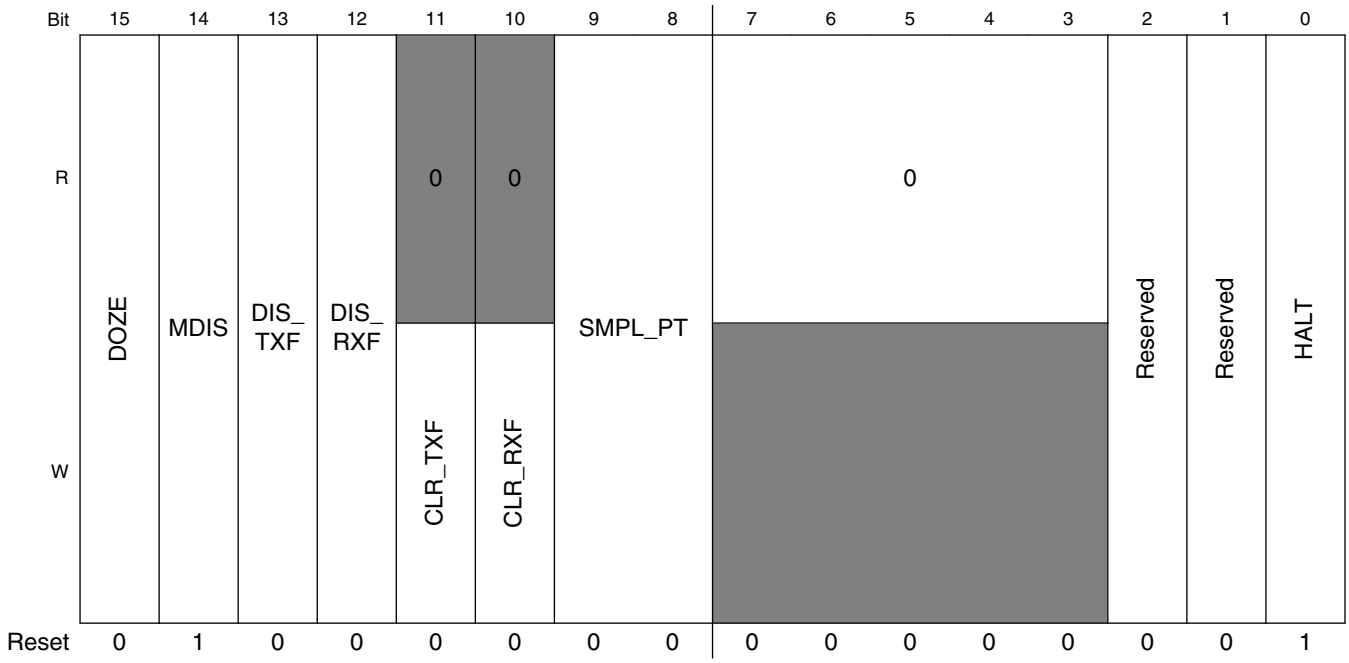
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_C00C	Clock and Transfer Attributes Register (In Slave Mode) (SPI0_CTAR0_SLAVE)	32	R/W	7800_0000h	40.3.4/1012
4002_C010	Clock and Transfer Attributes Register (In Master Mode) (SPI0_CTAR1)	32	R/W	7800_0000h	40.3.3/1007
4002_C02C	Status Register (SPI0_SR)	32	R/W	0200_0000h	40.3.5/1013
4002_C030	DMA/Interrupt Request Select and Enable Register (SPI0_RSER)	32	R/W	0000_0000h	40.3.6/1016
4002_C034	PUSH TX FIFO Register In Master Mode (SPI0_PUSHR)	32	R/W	0000_0000h	40.3.7/1018
4002_C034	PUSH TX FIFO Register In Slave Mode (SPI0_PUSHR_SLAVE)	32	R/W	0000_0000h	40.3.8/1020
4002_C038	POP RX FIFO Register (SPI0_POPR)	32	R	0000_0000h	40.3.9/1020
4002_C03C	Transmit FIFO Registers (SPI0_TXFR0)	32	R	0000_0000h	40.3.10/ 1021
4002_C040	Transmit FIFO Registers (SPI0_TXFR1)	32	R	0000_0000h	40.3.10/ 1021
4002_C044	Transmit FIFO Registers (SPI0_TXFR2)	32	R	0000_0000h	40.3.10/ 1021
4002_C048	Transmit FIFO Registers (SPI0_TXFR3)	32	R	0000_0000h	40.3.10/ 1021
4002_C07C	Receive FIFO Registers (SPI0_RXFR0)	32	R	0000_0000h	40.3.11/ 1021
4002_C080	Receive FIFO Registers (SPI0_RXFR1)	32	R	0000_0000h	40.3.11/ 1021
4002_C084	Receive FIFO Registers (SPI0_RXFR2)	32	R	0000_0000h	40.3.11/ 1021
4002_C088	Receive FIFO Registers (SPI0_RXFR3)	32	R	0000_0000h	40.3.11/ 1021

40.3.1 Module Configuration Register (SPIx_MCR)

Contains bits to configure various attributes associated with the module operations. The HALT and MDIS bits can be changed at any time, but the effect takes place only on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed, while the module is in the Running state.

Address: 4002_C000h base + 0h offset = 4002_C000h





SPIx_MCR field descriptions

Field	Description
31 MSTR	Master/Slave Mode Select Enables either Master mode (if supported) or Slave mode (if supported) operation. 0 Enables Slave mode 1 Enables Master mode
30 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0 Continuous SCK disabled. 1 Continuous SCK enabled.
29–28 DCONF	SPI Configuration. Selects among the different configurations of the module. 00 SPI 01 Reserved 10 Reserved 11 Reserved
27 FRZ	Freeze Enables transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in Debug mode. 1 Halt serial transfers in Debug mode.
26 MTFE	Modified Transfer Format Enable Enables a modified transfer format to be used.

Table continues on the next page...

SPIx_MCR field descriptions (continued)

Field	Description
	NOTE: When MTFE=1 with continuous SCK enabled (MCR [CONT_SCKE] =1) in master mode, configure CTAR[LSBFE]=0 for correct operations while receiving unequal length frames. If PUSHR[CONT] is also set for back to back frame transfer, also configure the frame size of the first frame as less than or equal to the frame size of the next frame. In this scenario, make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
25 Reserved	This field is reserved.
24 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the module to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
23–21 Reserved	Reserved. This field is reserved.
20–16 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx. Refer to the chip-specific SPI information for the number of PCS signals used in this chip. **NOTE:** The effect of this bit only takes place when module is enabled. Ensure that this bit is configured correctly before enabling the SPI interface. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.
15 DOZE	Doze Enable Provides support for an externally controlled Doze mode power-saving mechanism. 0 Doze mode has no effect on the module. 1 Doze mode disables the module.
14 MDIS	Module Disable Allows the clock to be stopped to the non-memory mapped logic in the module effectively putting it in a software-controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 1. When the module is used in Slave Mode, it is recommended to leave this bit 0, because a slave doesn't have control over master transactions. 0 Enables the module clocks. 1 Allows external logic to disable the module clocks.
13 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the module operates as a simplified double-buffered SPI. This bit can be written only when the MDIS bit is cleared.

Table continues on the next page...

SPIx_MCR field descriptions (continued)

Field	Description
	0 TX FIFO is enabled. 1 TX FIFO is disabled.
12 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the module operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 RX FIFO is enabled. 1 RX FIFO is disabled.
11 CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO counter. 1 Clear the TX FIFO counter.
10 CLR_RXF	CLR_RXF Clears the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. **NOTE:** After every RX FIFO clear operation (MCR [CLR_RXF] = 0b1) following a RX FIFO overflow (SR [RFOF] = 0b1) scenario, perform a single POP from the RX FIFO and discard the read data. The POP and discard operation must be completed before recieving a new incoming frame. 0 Do not clear the RX FIFO counter. 1 Clear the RX FIFO counter.
9–8 SMPL_PT	Sample Point Controls when the module master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTARn[CPHA] is 0. 00 0 protocol clock cycles between SCK edge and SIN sample 01 1 protocol clock cycle between SCK edge and SIN sample 10 2 protocol clock cycles between SCK edge and SIN sample 11 Reserved
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved.
1 Reserved	This field is reserved.
0 HALT	Halt The HALT bit starts and stops frame transfers. See Start and Stop of Module transfers 0 Start transfers. 1 Stop transfers.

40.3.2 Transfer Count Register (SPIx_TCR)

TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the module is in the Running state.

Address: 4002_C000h base + 8h offset = 4002_C008h

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	SPI_TCNT																0															
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	0	0	0

SPIx_TCR field descriptions

Field	Description
31–16 SPI_TCNT	SPI Transfer Counter Counts the number of SPI transfers the module makes. The SPI_TCNT field increments every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.3.3 Clock and Transfer Attributes Register (In Master Mode) (SPIx_CTAR*n*)

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

In Master mode, the CTAR registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bitfields in CTAR0 are used to set the slave transfer attributes.

When the module is configured as a SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTAR registers is used. When the module is configured as an SPI bus slave, it uses the CTAR0 register.

Address: 4002_C000h base + Ch offset + (4d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	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 W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPIx_CTAR*n* field descriptions

Field	Description
31 DBR	Double Baud Rate Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in the following table. See the BR field description for details on how to compute the baud rate.
30–27 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4.
26 CPOL	Clock Polarity

Table 40-2. SPI SCK Duty Cycle

DBR	CPHA	PBR	SCK Duty Cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

0 The baud rate is computed normally with a 50/50 duty cycle.
1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.

Table continues on the next page...

SPIx_CTAR*n* field descriptions (continued)

Field	Description
	Selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the module can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
25 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
24 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
23–22 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer PCS to SCK Delay (t_{CSC}) for more details. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
21–20 PASC	After SCK Delay Prescaler Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. Refer After SCK Delay (t_{ASC}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
19–18 PDT	Delay after Transfer Prescaler Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. See the DT field description for details on how to compute the Delay after Transfer. Refer Delay after Transfer (t_{DT}) for more details.

Table continues on the next page...

SPIx_CTAR*n* field descriptions (continued)

Field	Description
	00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
17–16 PBR	Baud Rate Prescaler Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
15–12 CSSCK	PCS to SCK Delay Scaler Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$. The following table lists the delay scaler values. **Table 40-3. Delay Scaler Encoding** **Field Value** / **Delay Scaler Value** 0000 / 2 0001 / 4 0010 / 8 0011 / 16 0100 / 32 0101 / 64 0110 / 128 0111 / 256 1000 / 512 1001 / 1024 1010 / 2048 1011 / 4096 1100 / 8192 1101 / 16384 1110 / 32768 1111 / 65536 Refer PCS to SCK Delay (t_{CSC}) for more details.
11–8 ASC	After SCK Delay Scaler

Table continues on the next page...

SPIx_CTAR*n* field descriptions (continued)

Field	Description
	Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{ASC} = (1/f_P) \times PASC \times ASC$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay (t_{ASC}) for more details.
7–4 DT	Delay After Transfer Scaler Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{DT} = (1/f_P) \times PDT \times DT$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values.
BR	Baud Rate Scaler Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation: SCK baud rate = $(f_P / PBR) \times [(1+DBR)/BR]$ The following table lists the baud rate scaler values.

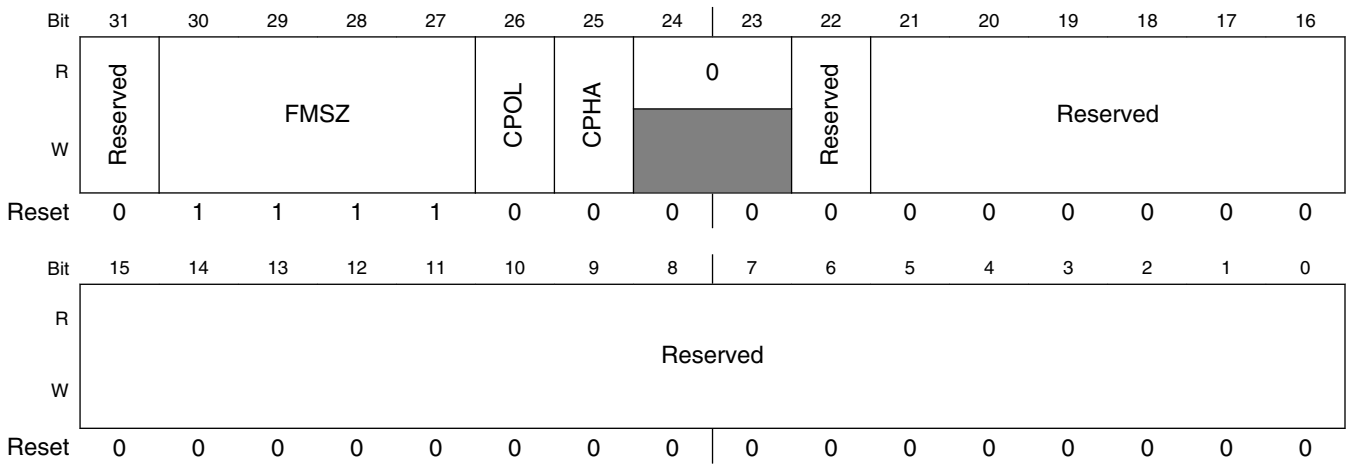
Table 40-4. Baud Rate Scaler

CTARn[BR]	Baud Rate Scaler Value
0000	2
0001	4
0010	6
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

40.3.4 Clock and Transfer Attributes Register (In Slave Mode) (SPIx_CTAR*n*_SLAVE)

When the module is configured as an SPI bus slave, the CTAR0 register is used.

Address: 4002_C000h base + Ch offset + (0d × i), where i=0d to 0d



SPIx_CTAR*n*_SLAVE field descriptions

Field	Description
31 Reserved	Always write the reset value to this field. This field is reserved.
30–27 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ field value plus 1. Note that the minimum valid value of frame size is 4.
26 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
25 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.

Table continues on the next page...

SPIx_CTAR*n*_SLAVE field descriptions (continued)

Field	Description
24–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved.
Reserved	This field is reserved.

40.3.5 Status Register (SPIx_SR)

SR contains status and flag bits. The bits reflect the status of the module and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the SR by writing a 1 to them. Writing a 0 to a flag bit has no effect. This register may not be writable in Module Disable mode due to the use of power saving mechanisms.

Address: 4002_C000h base + 2Ch offset = 4002_C02Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TCF	TXRXS	0	EOQF	TFUF	0	TFFF	0	0	0	0	0	RFOF	0	RFDF	Reserved
W	w1c			w1c	w1c		w1c						w1c		w1c	w1c
Reset	0	0	0	0	0	0	1	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	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPIx_SR field descriptions

Field	Description
31 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it. 0 Transfer not complete. 1 Transfer complete.
30 TXRXS	TX and RX Status Reflects the run status of the module. 0 Transmit and receive operations are disabled (The module is in Stopped state). 1 Transmit and receive operations are enabled (The module is in Running state).
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 EOQF	End of Queue Flag Indicates that the last entry in a queue has been transmitted when the module is in Master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.
27 TFUF	Transmit FIFO Underflow Flag Indicates an underflow condition in the TX FIFO. The transmit underflow condition is detected only for SPI blocks operating in Slave mode and SPI configuration. TFUF is set when the TX FIFO of the module operating in SPI Slave mode is empty and an external SPI master initiates a transfer. The TFUF bit remains set until cleared by writing 1 to it. 0 No TX FIFO underflow. 1 TX FIFO underflow has occurred.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 TFFF	Transmit FIFO Fill Flag Indicates whether there is an available location to be filled in the FIFO. Either a DMA request or an interrupt indication can be used to add another entry to the FIFO. Note that this bit is set if at least one location is free in the FIFO. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request. **NOTE:** The reset value of this bit is 0 when the module is disabled,(MCR[MDIS]=1). 0 TX FIFO is full. 1 TX FIFO is not full.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SPIx_SR field descriptions (continued)

Field	Description
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The field is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a 1 to it. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 RFDF	Receive FIFO Drain Flag Indicates whether there is an available location to be drained from the FIFO. Either a DMA request or an interrupt indication can be used to read from the FIFO. Note that this bit is set if at least one location can be read from the FIFO. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 RX FIFO is empty. 1 RX FIFO is not empty.
16 Reserved	This field is reserved.
15–12 TXCTR	TX FIFO Counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
11–8 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.
7–4 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

40.3.6 DMA/Interrupt Request Select and Enable Register (SPIx_RSER)

RSER controls DMA and interrupt requests. Do not write to the RSER while the module is in the Running state.

Address: 4002_C000h base + 30h offset = 4002_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TCF_RE	Reserved	Reserved	EOQF_RE	TFUF_RE	Reserved	TFFF_RE	TFFF_DIRS	Reserved	Reserved	Reserved	Reserved	RFOF_RE	Reserved	RFDF_RE	RFDF_DIRS
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	Reserved	Reserved	0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPIx_RSER field descriptions

Field	Description
31 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.
30 Reserved	Always write the reset value to this field. This field is reserved.
29 Reserved	Always write the reset value to this field. This field is reserved.
28 EOQF_RE	Finished Request Enable Enables the EOQF flag in the SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
27 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the SR to generate an interrupt request.

Table continues on the next page...

SPIx_RSER field descriptions (continued)

Field	Description
	0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.
26 Reserved	Always write the reset value to this field. This field is reserved.
25 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 TFFF interrupts or DMA requests are disabled. 1 TFFF interrupts or DMA requests are enabled.
24 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When SR[TFFF] and RSER[TFFF_RE] are set, this field selects between generating an interrupt request or a DMA request. 0 TFFF flag generates interrupt requests. 1 TFFF flag generates DMA requests.
23 Reserved	Always write the reset value to this field. This field is reserved.
22 Reserved	Always write the reset value to this field. This field is reserved.
21 Reserved	Always write the reset value to this field. This field is reserved.
20 Reserved	Always write the reset value to this field. This field is reserved.
19 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
18 Reserved	Always write the reset value to this field. This field is reserved.
17 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt or DMA requests are disabled. 1 RFDF interrupt or DMA requests are enabled.
16 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select

Table continues on the next page...

SPIx_RSER field descriptions (continued)

Field	Description
	Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
15 Reserved	Always write the reset value to this field. This field is reserved.
14 Reserved	Always write the reset value to this field. This field is reserved.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.3.7 PUSH TX FIFO Register In Master Mode (SPIx_PUSHR)

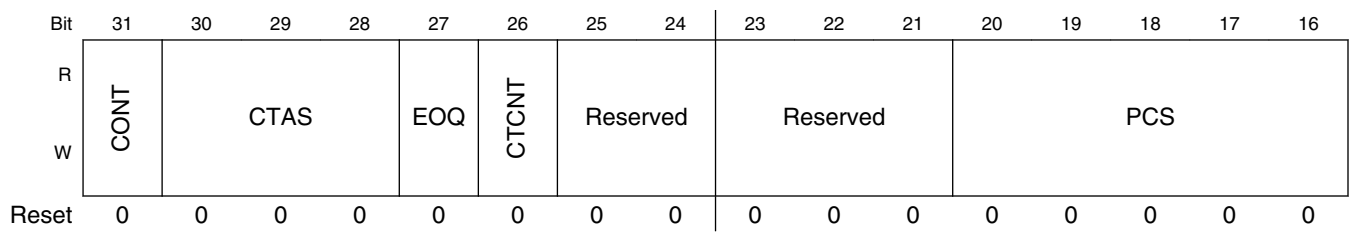
Specifies data to be transferred to the TX FIFO and CMD FIFO. User must write 16-bits data into TXDATA field. An 8- or 16-bit write access to the TXDATA field transfers 16 bits of data bus to the TX FIFO. A write access to the command fields transfers the 16 bits of command information to the CMD FIFO. In Master mode, the register transfers 16 bits of data to the TX FIFO and 16 bits of command information to the CMD FIFO.

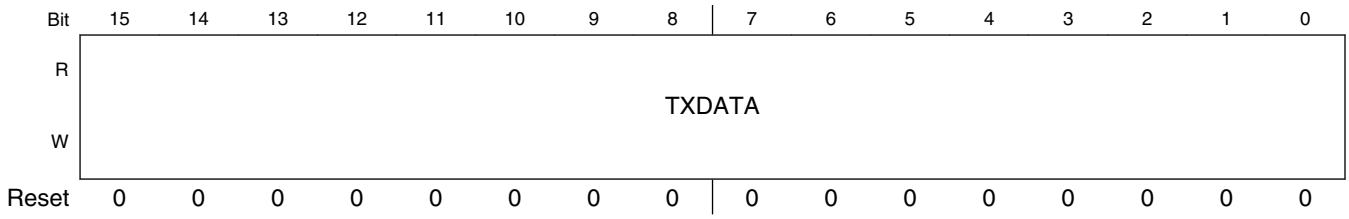
The TX FIFO and CMD FIFO must be filled simultaneously. In other words, you must perform write accesses to both the data and command fields for every PUSHR operation. Because both the TX FIFO and CMD FIFO are written to and read from simultaneously, they behave as a single 32 bit FIFO.

A read access of PUSHR returns the topmost TX FIFO and CMD FIFO entries concatenated.

When the module is disabled, writing to this register does not update the FIFO. Therefore, any reads performed while the module is disabled return the last PUSHR write performed while the module was still enabled.

Address: 4002_C000h base + 34h offset = 4002_C034h





SPIx_PUSHR field descriptions

Field	Description
31 CONT	Continuous Peripheral Chip Select Enable Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
30–28 CTAS	Clock and Transfer Attributes Select Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI Slave mode, CTAR0 is used. See the chip specific section for details to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000 CTAR0 001 CTAR1 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
27 EOQ	End Of Queue Host software uses this bit to signal to the module that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
26 CTCNT	Clear Transfer Counter Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame. 0 Do not clear the TCR[TCNT] field. 1 Clear the TCR[TCNT] field.
25–24 Reserved	Always write the reset value to this field. This field is reserved.
23–21 Reserved	Always write the reset value to this field. This field is reserved.
20–16 PCS	Select which PCS signals are to be asserted for the transfer. Refer to the chip-specific SPI information for the number of PCS signals used in this chip.

Table continues on the next page...

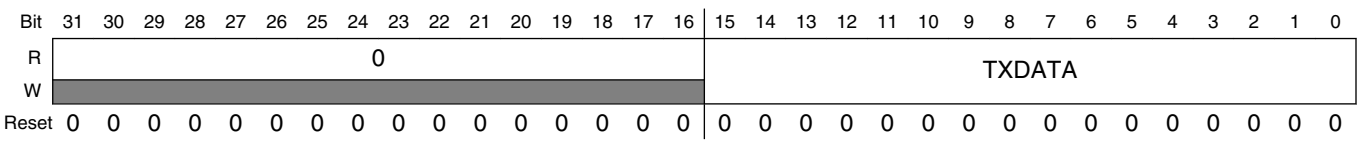
SPIx_PUSHR field descriptions (continued)

Field	Description
	0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

40.3.8 PUSH TX FIFO Register In Slave Mode (SPIx_PUSHR_SLAVE)

Specifies data to be transferred to the TX FIFO in slave mode. An 8- or 16-bit write access to PUSHR transfers the 16-bit TXDATA field to the TX FIFO.

Address: 4002_C000h base + 34h offset = 4002_C034h



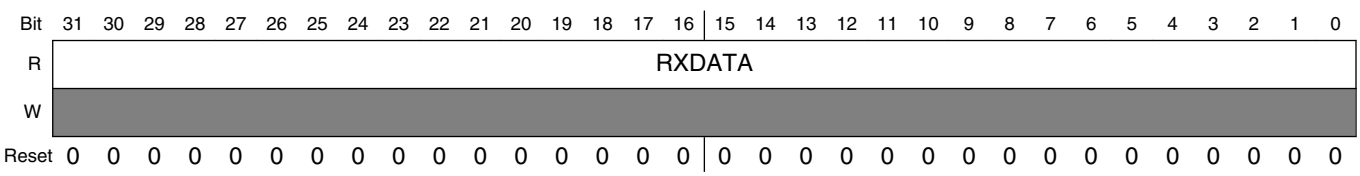
SPIx_PUSHR_SLAVE field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

40.3.9 POP RX FIFO Register (SPIx_POPR)

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

Address: 4002_C000h base + 38h offset = 4002_C038h



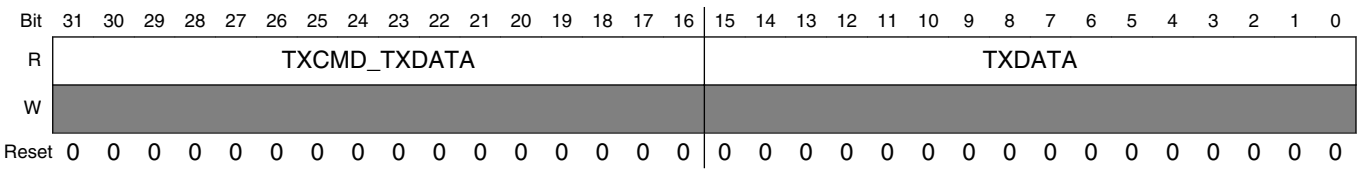
SPIx_POPR field descriptions

Field	Description
RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

40.3.10 Transmit FIFO Registers (SPIx_TXFR*n*)

TXFRn registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRx registers does not alter the state of the TX FIFO.

Address: 4002_C000h base + 3Ch offset + (4d × i), where i=0d to 3d



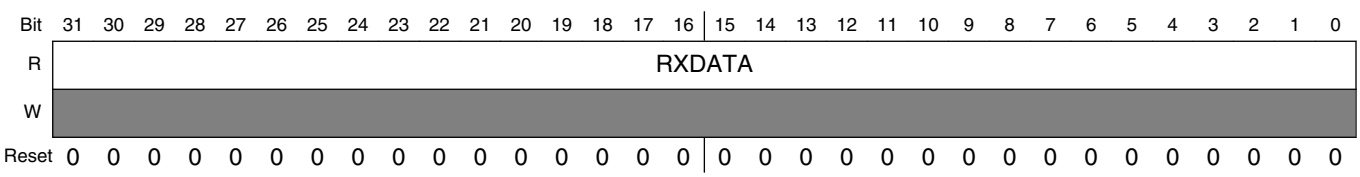
SPIx_TXFR*n* field descriptions

Field	Description
31–16 TXCMD_ TXDATA	Transmit Command or Transmit Data In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data. In Slave mode, this field is reserved.
TXDATA	Transmit Data Contains the SPI data to be shifted out.

40.3.11 Receive FIFO Registers (SPIx_RXFR*n*)

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFR registers are read-only. Reading the RXFRx registers does not alter the state of the RX FIFO.

Address: 4002_C000h base + 7Ch offset + (4d × i), where i=0d to 3d



SPIx_RXFR*n* field descriptions

Field	Description
RXDATA	Receive Data Contains the received SPI data.

40.4 Functional description

The module supports full-duplex, synchronous serial communications between chips and peripheral devices. The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes.

The module has the following configuration

- The SPI Configuration in which the module operates as a basic SPI or a queued SPI.

The DCONF field in the Module Configuration Register (MCR) determines the module Configuration. SPI configuration is selected when DCONF within SPIx_MCR is 0b00.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command.

See Clock and Transfer Attributes Register (In Master Mode) (SPI_CTAR*n*) for information on the fields of CTAR registers.

Typical master to slave connections are shown in the following figure. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the Transfer Control Flag(TCF) bit in the Shift Register(SR) is set to indicate a completed frame transfer.

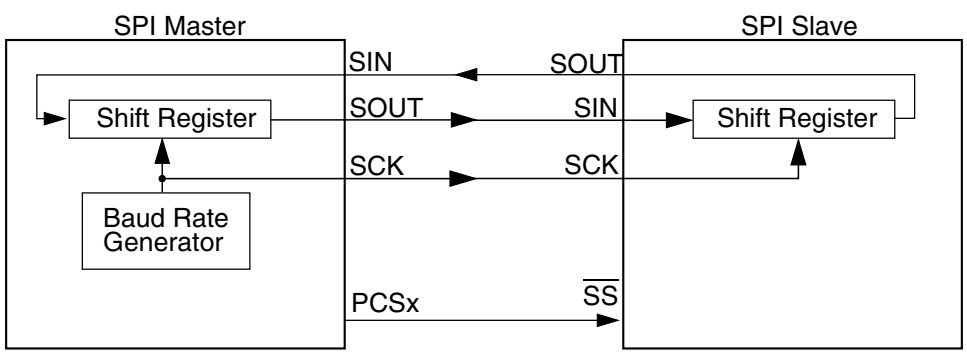


Figure 40-3. Serial protocol overview

Generally, more than one slave device can be connected to the module master. 5 Peripheral Chip Select (PCS) signals of the module masters can be used to select which of the slaves to communicate with. Refer to the chip specific section for details on the number of PCS signals used in this chip.

The SPI configuration shares transfer protocol and timing properties which are described independently of the configuration in Transfer formats. The transfer rate and delay settings are described in Module baud rate and clock delay generation.

40.4.1 Start and Stop of module transfers

The module has two operating states: Stopped and Running. Both the states are independent of it's configuration. The default state of the module is Stopped. In the Stopped state, no serial transfers are initiated in Master mode and no transfers are responded to in Slave mode. The Stopped state is also a safe state for writing the various configuration registers of the module without causing undetermined results. In the Running state serial transfers take place.

The TXRXS bit in the SR indicates the state of module. The bit is set if the module is in Running state.

The module starts or transitions to Running when all of the following conditions are true:

- SR[EOQF] bit is clear
- Chip is not in the Debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The module stops or transitions from Running to Stopped after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- Chip in the Debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

40.4.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The module is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to the module RAM queues to a TX FIFO buffer. The received data is stored in entries in the RX FIFO buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the module. The operation of FIFO buffers is described in the following sections:

- Transmit First In First Out (TX FIFO) buffering mechanism
- Command First In First Out (CMD FIFO) Buffering Mechanism
- Receive First In First Out (RX FIFO) buffering mechanism

The interrupt and DMA request conditions are described in Interrupts/DMA requests.

The SPI configuration supports two block-specific modes—Master mode and Slave mode. In Master mode the module initiates and controls the transfer according to the fields of the executing SPI Command. In Slave mode, the module responds only to transfers initiated by a bus master external to it and the SPI command field space is reserved.

40.4.2.1 Master mode

In SPI Master mode, the module initiates the serial transfers by controlling the SCK and the PCS signals. The executing SPI Command determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See PUSH TX FIFO Register In Master Mode (SPI_PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame by frame basis.

40.4.2.2 Slave mode

In SPI Slave mode the module responds to transfers initiated by an SPI bus master. It does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase, and frame size must be set for successful communication with an SPI master. The SPI Slave mode transfer attributes are set in the CTAR0. The data is shifted out with MSB first. Shifting out of LSB is not supported in this mode.

40.4.2.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO. The module operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately. Setting the MCR[DIS_TXF] bit disables the TX FIFO and CMD FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO and CMD FIFO are disabled:
- SR[TFFF], SR[TFUF] and SR[TXCTR] behave as if there is a one-entry FIFO
- The contents of TXFRs, SR[TXNXTPTR] are undefined

Similarly, when the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

40.4.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds 4 words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data Field of module PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the module Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time a 8- or 16-bit write takes place to PUSHR[TXDATA] or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates the TX FIFO Entry that will be transmitted during the next transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR number and it rolls over after reaching the maximum.

40.4.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA

controller indicates that a write to PUSHR is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See Transmit FIFO Fill Interrupt or DMA Request for details.

The module ignores attempts to push data to a full TX FIFO, and the state of the TX FIFO does not change and no error condition is indicated.

40.4.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a module slave while the slave's TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See Transmit FIFO Underflow Interrupt Request for details.

40.4.2.5 Command First In First Out (CMD FIFO) Buffering Mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission. The TX FIFO and CMD FIFO must be filled together, i.e. write enables should be given for both the Data and Command fields while performing a PUSHR operation.

The CMD FIFO holds 4 words, each representing SPI command fields. The number of entries in the CMD FIFO is device-specific. SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

Every CMD FIFO entry has a corresponding single TX FIFO entry attached to it because both these FIFO's are filled simultaneously.

40.4.2.6 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds 4 received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the module POP RX FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the module's Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in a number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when the POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR number and it rolls over after reaching the maximum.

40.4.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO, the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

40.4.2.6.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the module POP RX FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a 1 to it.

40.4.3 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the system clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

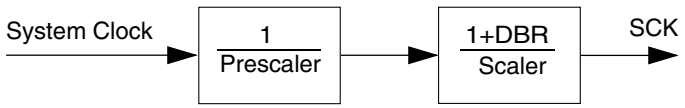


Figure 40-4. Communications clock prescalers and scalers

40.4.3.1 Baud rate generator

The baud rate is the frequency of the SCK. The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR, and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

Table 40-5. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

NOTE

The clock frequencies mentioned in the preceding table are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

40.4.3.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 40-5 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR*x* registers select the PCS to SCK delay by the formula in the CSSCK field description. The following table shows an example of how to compute the PCS to SCK delay.

Table 40-6. PCS to SCK delay computation example

f_{SYS}	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

40.4.3.3 After SCK Delay (t_{ASC})

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 40-5 and Figure 40-6 for illustrations of the After SCK delay. The PASC and ASC fields in the CTAR*x* registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

Table 40-7. After SCK Delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

40.4.3.4 Delay after Transfer (t_{DT})

The Delay after Transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 40-5 for an illustration of the Delay after Transfer. The PDT and DT fields in the CTAR*x* registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the Delay after Transfer.

Table 40-8. Delay after Transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after Transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in Non-Continuous Clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] field description. When in Continuous Clock mode, the delay is fixed at 1 SCK period.

40.4.4 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In Master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL - Selects the idle state polarity of the SCK
- CPHA - Selects if the data on SOUT is valid before or on the first SCK edge

Even though the bus slave does not control the SCK signal, in Slave mode the values of CPOL and CPHA must be identical to the master device settings to ensure proper transmission. In SPI Slave mode, only CTAR0 is used.

The module supports four different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer Format with CPHA = 0
- Modified Transfer Format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The module can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the interface configurations, the module provides the option of keeping the PCS signals asserted between frames. See Continuous Selection Format for details.

40.4.4.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

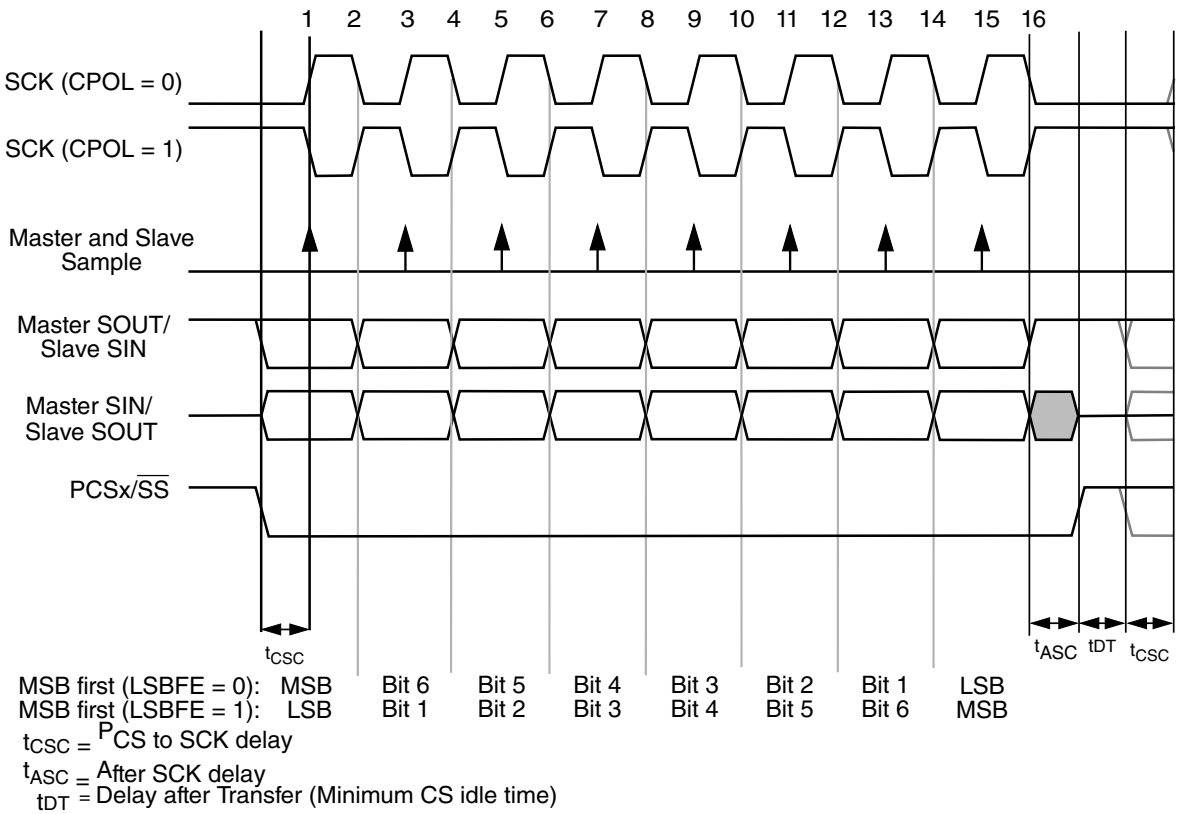


Figure 40-5. Module transfer timing diagram (MTFE=0, CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK, the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

40.4.4.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

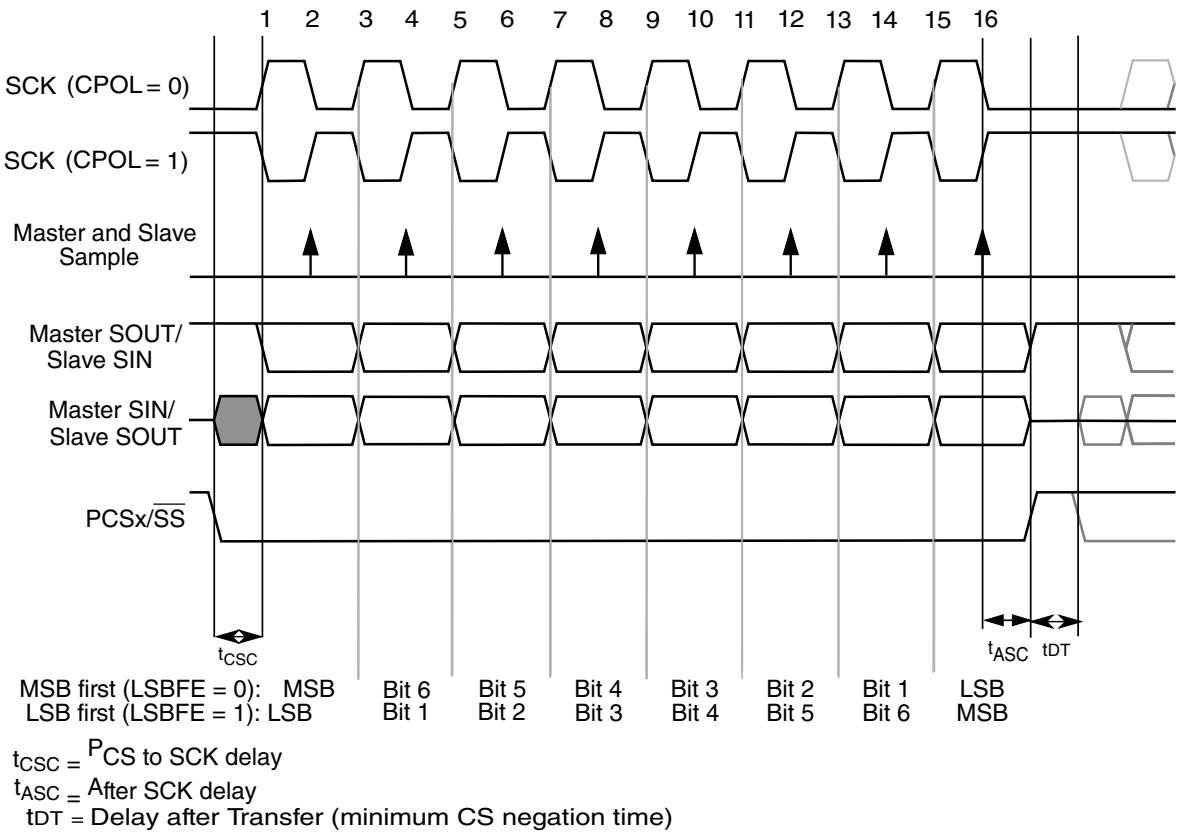


Figure 40-6. Module transfer timing diagram (MTFE=0, CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

40.4.4.3 Modified SPI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actual sample point is delayed by one protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE=1. Timing delays shown are:

- T_{csc} - PCS to SCK assertion delay
- T_{acs} - After SCK PCS negation delay
- T_{su_ms} - master SIN setup time
- T_{hd_ms} - master SIN hold time
- T_{vd_sl} - slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- T_{su_sl} - data setup time on slave data input
- T_{hd_sl} - data hold time on slave data input
- T_{sys} - protocol clock period.

The following figure shows the modified transfer format for CPHA = 0 and Fsys/Fsck = 4. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behavior are shown.

- Signal, marked "SOUT of Ext Slave", presents regular SPI slave serial output.
- Signal, marked "SOUT of DSPI Slave", presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note

In the following diagrams, f_{sys} represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

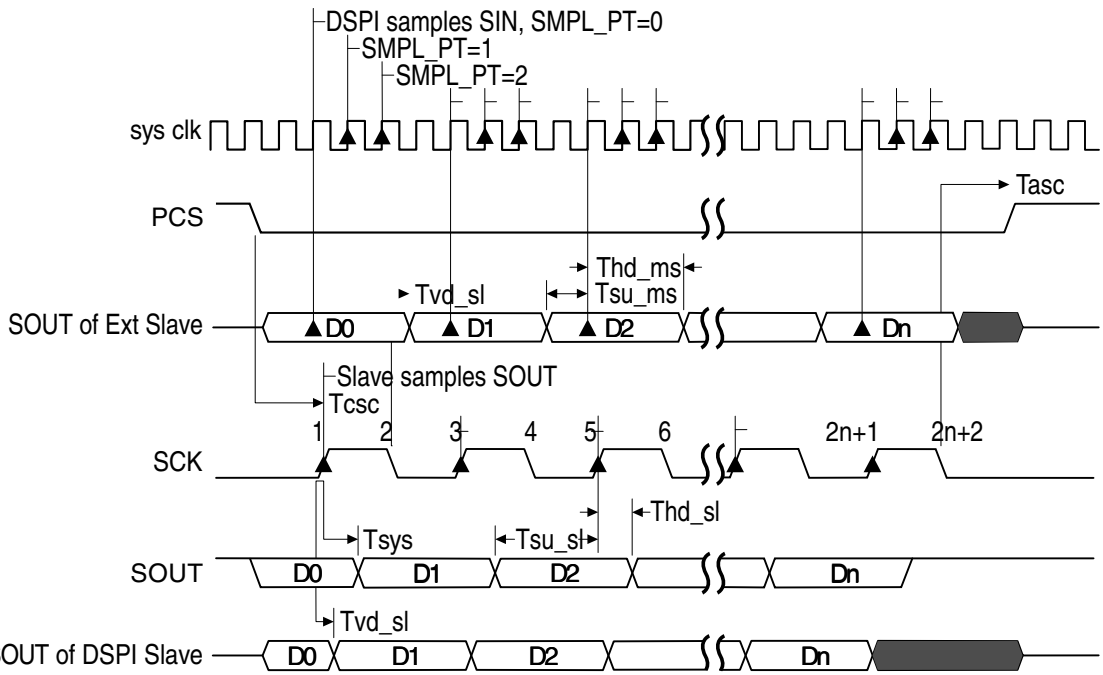


Figure 40-7. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/4$)

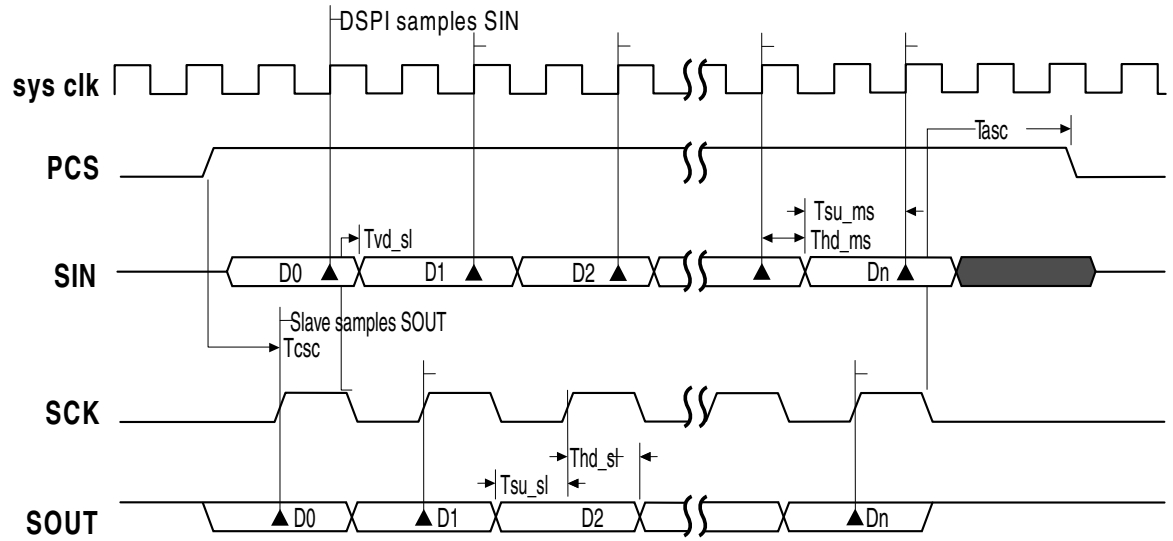


Figure 40-8. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/2$)

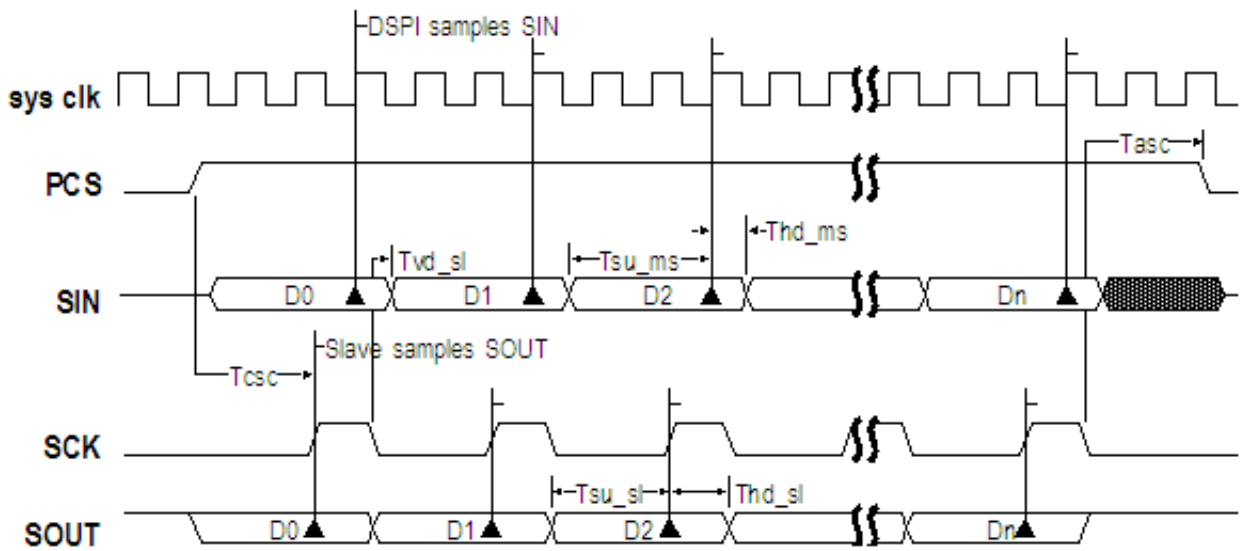


Figure 40-9. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/3$)

40.4.4.4 Modified SPI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK . The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT

signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The SCK to PCS delay and the After SCK delay must be greater or equal to half of the SCK period.**

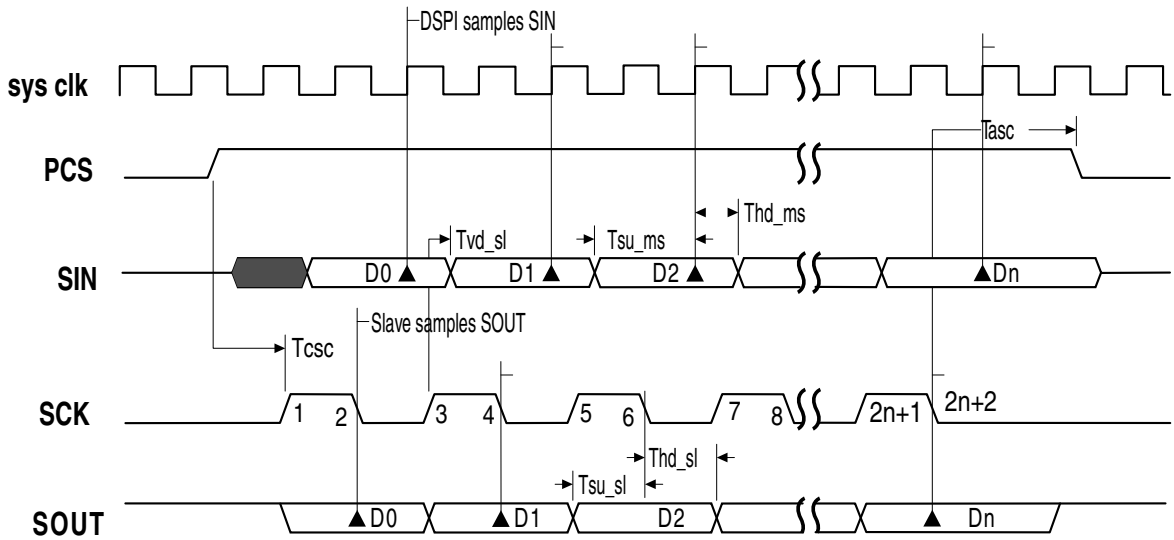


Figure 40-10. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/2$)

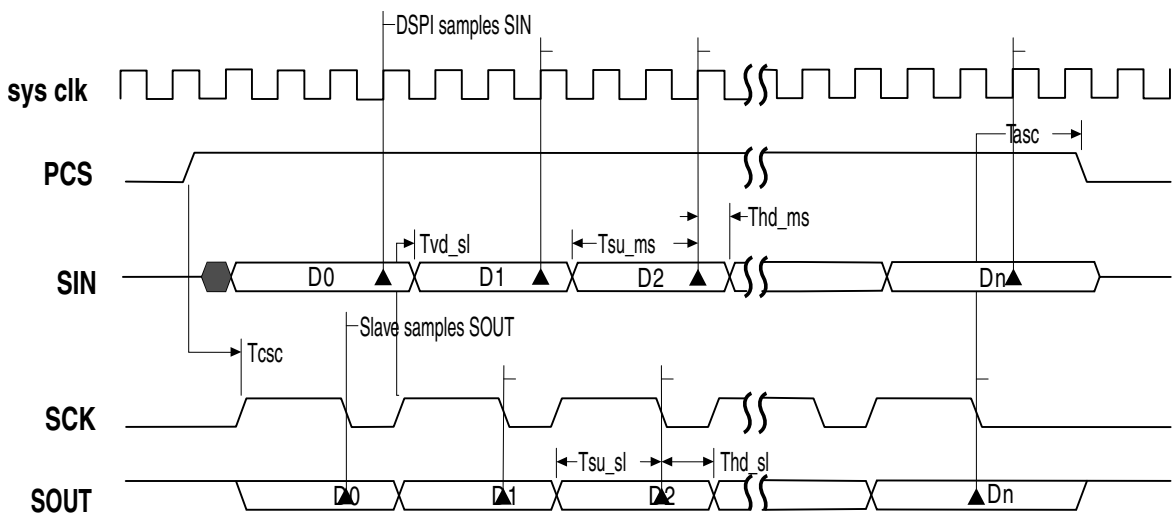


Figure 40-11. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/3$)

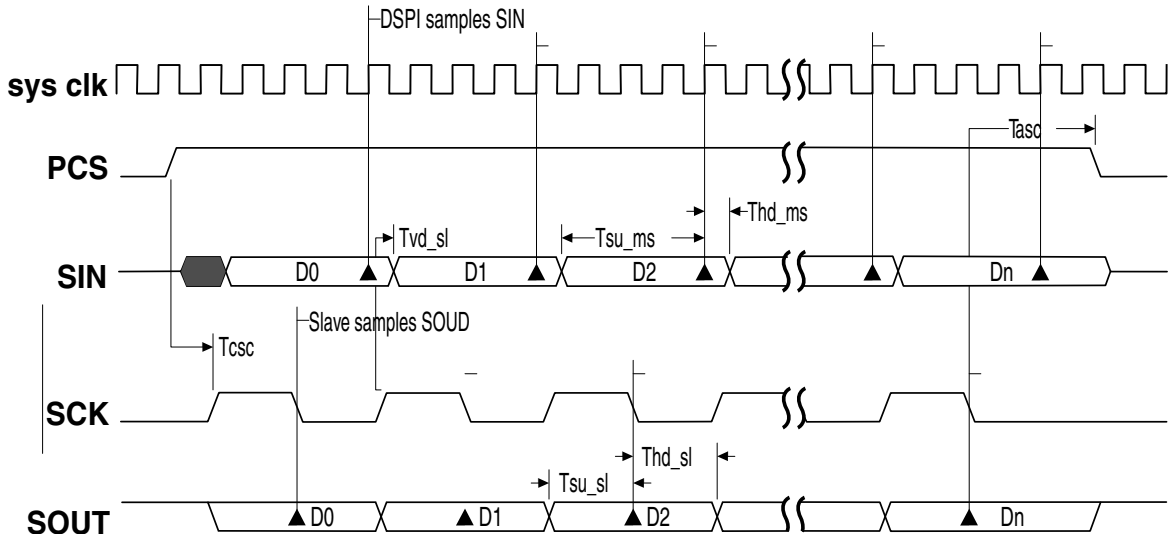


Figure 40-12. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/4$)

40.4.4.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle the following case. The Continuous Selection Format is enabled for the SPI configuration by setting the CONT bit in the SPI command.

When the CONT bit = 0, the module drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The following timing diagram is for two four-bit transfers with CPHA = 1 and CONT = 0.

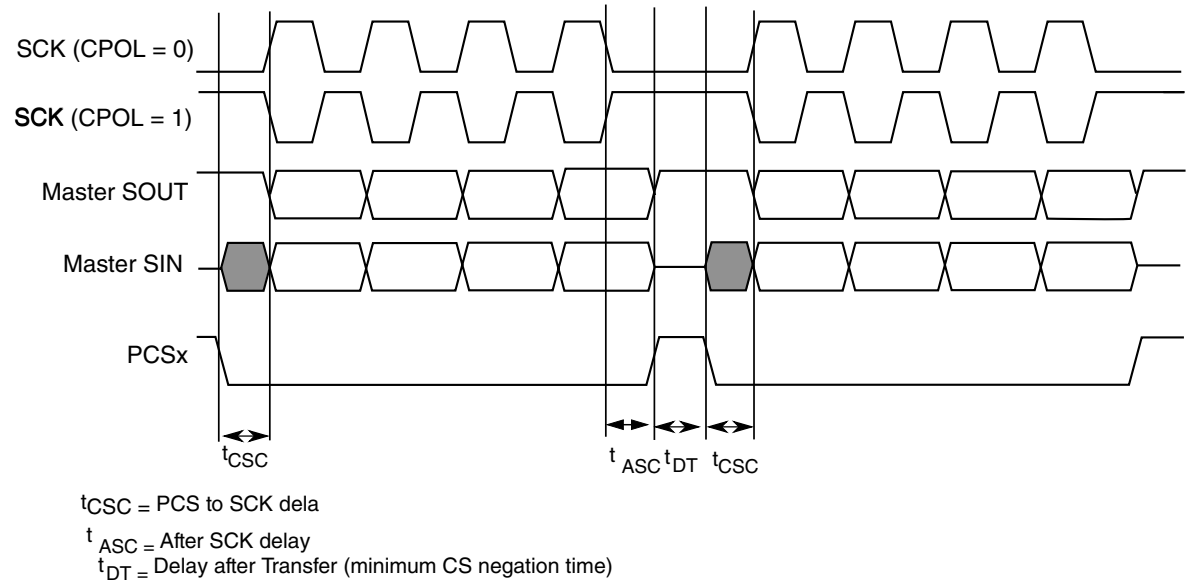


Figure 40-13. Example of non-continuous format (CPHA=1, CONT=0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. The following figure shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

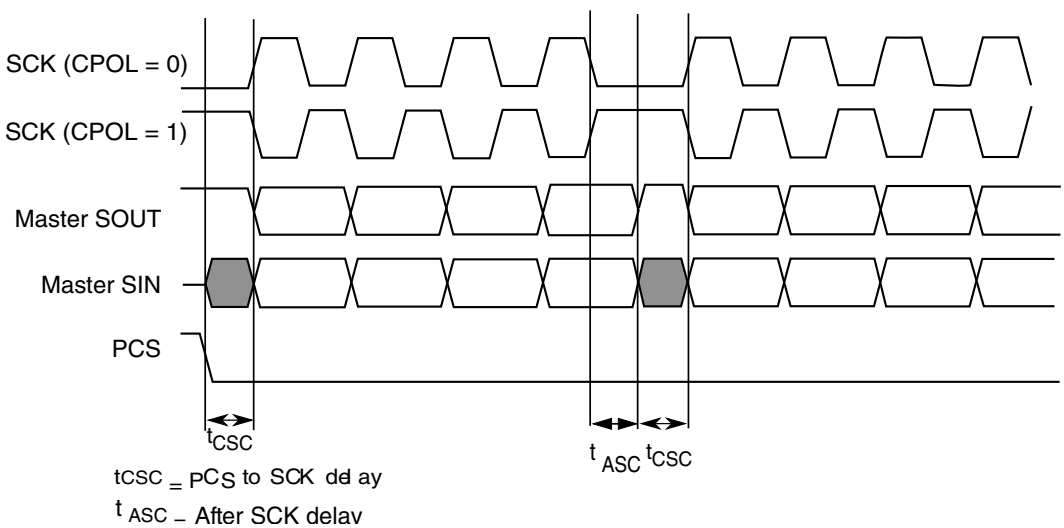


Figure 40-14. Example of continuous transfer (CPHA=1, CONT=1)

When using the module with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.

- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit deasserted in Master mode and the user software must provide sufficient frames in the TX_FIFO to be sent out in Slave mode and the master deasserts the PCSn at end of transmission of the last frame.
- PUSHR[CONT] must be deasserted before asserting MCR[HALT] in master mode. This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from Running to Stopped state.

NOTE

User must fill the TX FIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TX FIFO becomes empty.

When operating in Slave mode, ensure that when the last entry in the TX FIFO is completely transmitted, that is, the corresponding TCF flag is asserted and TXFIFO is empty, the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

40.4.5 Continuous Serial Communications Clock

The module provides the option of generating a Continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA=1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the module is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. The following figure is the timing diagram for Continuous SCK format with Continuous Selection disabled.

NOTE

In Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TX FIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

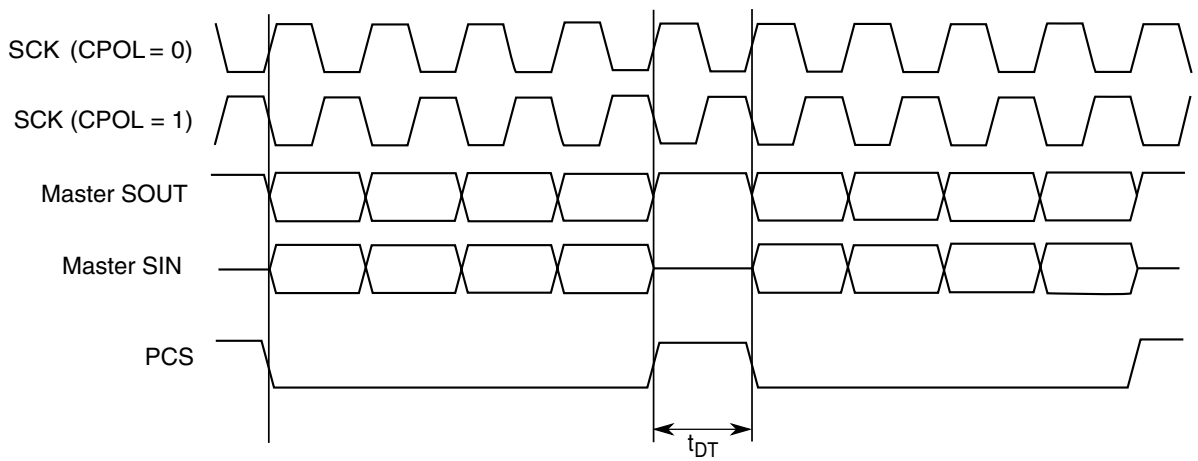


Figure 40-15. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT, that is, SOUT pulled high. This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the TX FIFO.
- Continuous SCK with CONT bit set and entering Stopped state (refer to Start and Stop of module transfers).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

The following figure shows timing diagram for Continuous SCK format with Continuous Selection enabled.

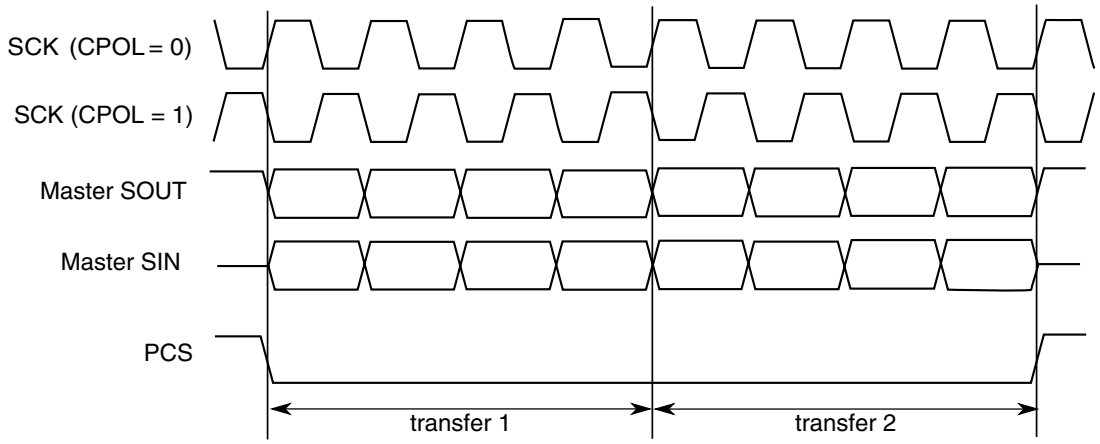


Figure 40-16. Continuous SCK timing diagram (CONT=1)

40.4.6 Slave Mode Operation Constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the module is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and $\overline{SS}$ signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

40.4.7 Interrupts/DMA requests

The module has several conditions that can generate only interrupt requests and two conditions that can generate interrupt or DMA requests. The following table lists these conditions.

Table 40-9. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	-
TX FIFO Fill	TFFF	Yes	Yes
Transfer Complete	TCF	Yes	-
TX FIFO Underflow	TFUF	Yes	-

Table continues on the next page...

Table 40-9. Interrupt and DMA request conditions (continued)

Condition	Flag	Interrupt	DMA
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	-

Each condition has a flag bit in the module Status Register (SR) and a Request Enable bit in the DMA/Interrupt Request Select and Enable Register (RSER). Certain flags (as shown in above table) generate interrupt requests or DMA requests depending on configuration of RSER register.

The module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

40.4.7.1 End Of Queue interrupt request

The End Of Queue (EOQ) interrupt request indicates that the end of a transmit queue is reached. The module generates the interrupt request when EOQ interrupt requests are enabled (RSER[EOQF_RE]) and the EOQ bit in the executing SPI command is 1.

The module generates the interrupt request when the last bit of the SPI frame with EOQ bit set is transmitted.

40.4.7.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

NOTE

TFFF flag clears automatically when DMA is used to fill TX FIFO. Configure the DMA to fill only one FIFO location per transfer.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TX FIFO:

1. Wait until TFFF = 1.
2. Write data to PUSHR using CPU.

3. Clear TFFF by writing a 1 to its location. If TX FIFO is not full, this flag will not clear.

40.4.7.3 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

40.4.7.4 Transmit FIFO Underflow Interrupt Request

The Transmit FIFO Underflow Request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the module operating in Slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of the module is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

40.4.7.5 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated. Configure the DMA to drain only one FIFO location per transfer.

40.4.7.6 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

40.4.8 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

40.4.8.1 Stop mode (External Stop mode)

This module supports the Stop mode protocol. When a request is made to enter External Stop mode, the module acknowledges the request . If a serial transfer is in progress, then this module waits until it reaches the frame boundary before it is ready to have its clocks shut off . While the clocks are shut off, this module's memory-mapped logic is not accessible. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

40.4.8.2 Module Disable mode

Module Disable mode is a block-specific mode that the module can enter to save power. Host CPU can initiate the Module Disable mode by setting the MDIS bit in the MCR. The Module Disable mode can also be initiated by hardware.

When the MDIS bit is set, the module negates the Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, it is said to have entered Module Disable Mode. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. If implemented, the Clock Enable signal can stop the clock to the non-memory mapped logic. When Clock Enable is negated, the module is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the module is in the Module Disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Similarly, writing to the PUSHR Register does not change the state of the TX FIFO or CMD FIFO. Clearing either of the FIFOs has no effect in the Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the module return the correct values when read, but writing to them has no effect. Writing to the TCR during Module Disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

40.5 Initialization/application information

This section describes how to initialize the module.

40.5.1 How to manage queues

The queues are not part of the module, but it includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When module executes last command word from a queue, the EOQ bit in the command word is set to indicate it that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the module in the Stopped state. The TXRXS bit is cleared to indicate the Stopped state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a 1 to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the module TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

40.5.2 Switching Master and Slave mode

When changing modes in the module, follow the steps below to guarantee proper operation.

1. Halt it by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a 1 to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable it by clearing MCR[HALT].

40.5.3 Initializing Module in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the module is enabled by clearing MCR[HALT]. It should be ensured that module Slave is enabled before enabling it's Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

40.5.4 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is cleared.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

Table 40-10. Baud rate values (bps)

		Baud rate divider prescaler values			
		2	3	5	7
Baud Rate Scaler Values	2	25.0M	16.7M	10.0M	7.14M
	4	12.5M	8.33M	5.00M	3.57M
	6	8.33M	5.56M	3.33M	2.38M
	8	6.25M	4.17M	2.50M	1.79M
	16	3.12M	2.08M	1.25M	893k
	32	1.56M	1.04M	625k	446k
	64	781k	521k	312k	223k
	128	391k	260k	156k	112k
	256	195k	130k	78.1k	55.8k
	512	97.7k	65.1k	39.1k	27.9k
	1024	48.8k	32.6k	19.5k	14.0k
	2048	24.4k	16.3k	9.77k	6.98k
	4096	12.2k	8.14k	4.88k	3.49k
	8192	6.10k	4.07k	2.44k	1.74k
	16384	3.05k	2.04k	1.22k	872
	32768	1.53k	1.02k	610	436

40.5.5 Delay settings

The following table shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (T_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

Table 40-11. Delay values

		Delay prescaler values			
		1	3	5	7
Delay scaler values	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

40.5.6 Calculation of FIFO pointer addresses

Complete visibility of the FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the CMD FIFO the first-in pointer is the Command Next Pointer (CMDNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). The following figure illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over. See Transmit First In First Out (TX FIFO) buffering mechanism, Command First In First Out (CMD FIFO) Buffering Mechanism and Receive First In First Out (RX FIFO) buffering mechanism for details on the FIFO operation.

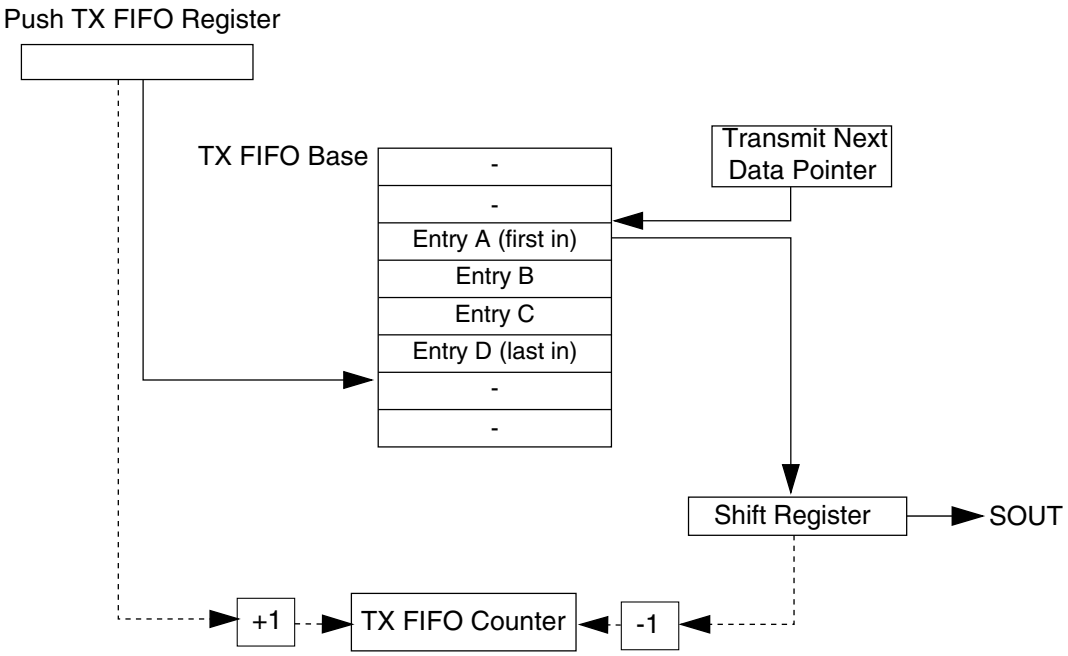


Figure 40-17. TX FIFO pointers and counter

40.5.6.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + \left(4 \times \text{TXNXTPTR}\right)$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times \left(\text{TXCTR} + \text{TXNXTPTR} - 1\right) \text{mod} \left(\text{TXFIFOdepth}\right)$$

TX FIFO Base - Base address of TX FIFO
TXCTR - TX FIFO Counter
TXNXTPTR - Transmit Next Pointer
TX FIFO Depth - Transmit FIFO depth, implementation specific

40.5.6.2 Address Calculation for the First-in Entry and Last-in Entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFOdepth})$$

CMD FIFO Base - Base address of CMD FIFO
CMDCTR - CMD FIFO Counter
CMDNXTPTR - Command Next Pointer
CMD FIFO Depth - Command FIFO depth, implementation specific

40.5.6.3 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + (4 \times \text{POPNXTPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times (\text{RXCTR} + \text{POPNXTPTR} - 1) \bmod (\text{RXFIFOdepth})$$

RX FIFO Base - Base address of RX FIFO
RXCTR - RX FIFO counter
POPNXTPTR - Pop Next Pointer
RX FIFO Depth - Receive FIFO depth, implementation specific

Chapter 41
Inter-Integrated Circuit (I2C)

41.1 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*.

41.1.1 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 input glitch filter
- Low power mode wakeup on slave address match
- Range slave address support
- DMA support

41.1.2 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.

41.1.3 Block diagram

The following figure is a functional block diagram of the I2C module.

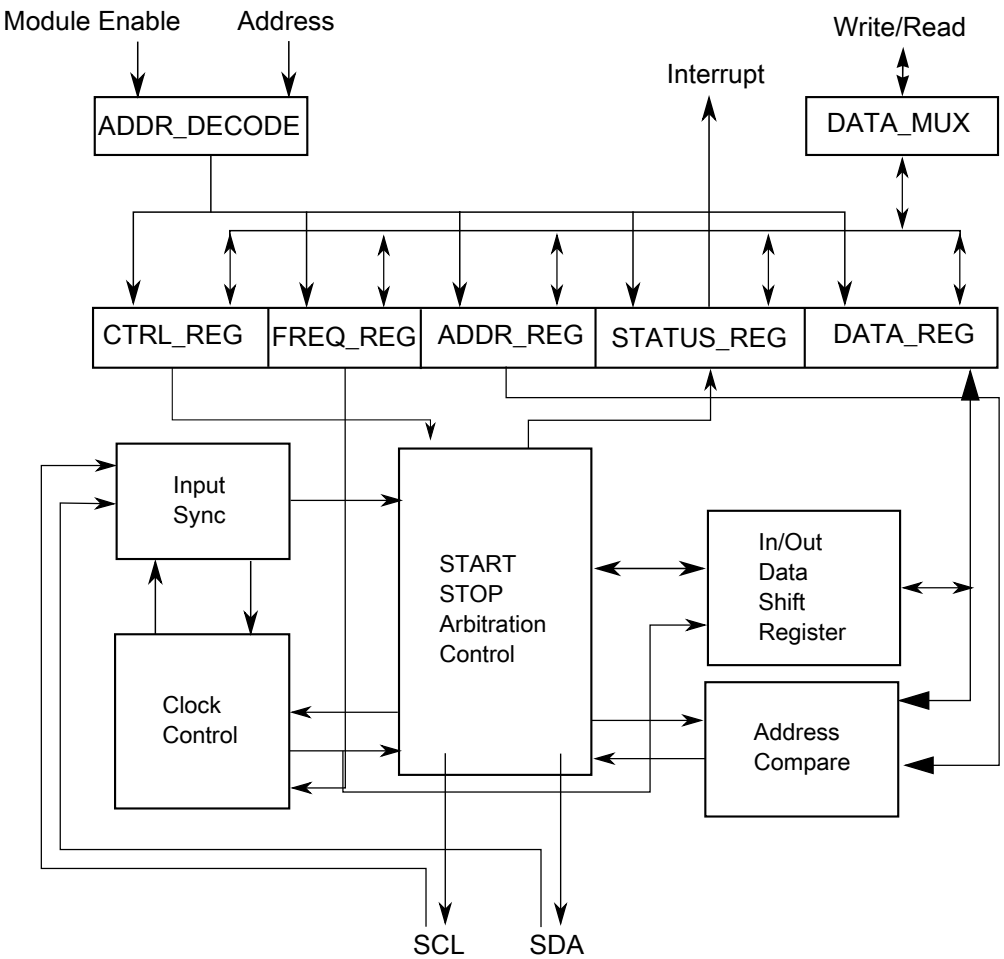


Figure 41-1. I2C Functional block diagram

41.2 I²C signal descriptions

The signal properties of I²C are shown in the table found here.

Table 41-1. 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

41.3 Memory map/register definition

This section describes in detail all I2C registers accessible to the end user.

I2C memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_6000	I2C Address Register 1 (I2C0_A1)	8	R/W	00h	41.3.1/1054
4006_6001	I2C Frequency Divider register (I2C0_F)	8	R/W	00h	41.3.2/1055
4006_6002	I2C Control Register 1 (I2C0_C1)	8	R/W	00h	41.3.3/1056
4006_6003	I2C Status register (I2C0_S)	8	R/W	80h	41.3.4/1057
4006_6004	I2C Data I/O register (I2C0_D)	8	R/W	00h	41.3.5/1059
4006_6005	I2C Control Register 2 (I2C0_C2)	8	R/W	00h	41.3.6/1060
4006_6006	I2C Programmable Input Glitch Filter Register (I2C0_FLT)	8	R/W	00h	41.3.7/1061
4006_6007	I2C Range Address register (I2C0_RA)	8	R/W	00h	41.3.8/1062
4006_6008	I2C SMBus Control and Status register (I2C0_SMB)	8	R/W	00h	41.3.9/1063
4006_6009	I2C Address Register 2 (I2C0_A2)	8	R/W	C2h	41.3.10/ 1065
4006_600A	I2C SCL Low Timeout Register High (I2C0_SLTH)	8	R/W	00h	41.3.11/ 1065
4006_600B	I2C SCL Low Timeout Register Low (I2C0_SLTL)	8	R/W	00h	41.3.12/ 1065

41.3.1 I2C Address Register 1 (I2Cx_A1)

This register contains the slave address to be used by the I2C module.

Address: 4006_6000h base + 0h offset = 4006_6000h

Bit	7	6	5	4	3	2	1	0
Read	AD[7:1]							0
Write								
Reset	0	0	0	0	0	0	0	0

I2Cx_A1 field descriptions

Field	Description
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.

41.3.2 I2C Frequency Divider register (I2Cx_F)

Address: 4006_6000h base + 1h offset = 4006_6001h

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

I2Cx_F field descriptions

Field	Description
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
	
	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

Table continues on the next page...

I2Cx_F field descriptions (continued)

Field	Description				
	MULT	ICR	Hold times (µs)		
			SDA	SCL Start	SCL Stop
	0h	18h	1.125	4.750	5.125

41.3.3 I2C Control Register 1 (I2Cx_C1)

Address: 4006_6000h base + 2h offset = 4006_6002h

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
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

Table continues on the next page...

I2Cx_C1 field descriptions (continued)

Field	Description
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.
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.

41.3.4 I2C Status register (I2Cx_S)

Address: 4006_6000h base + 3h offset = 4006_6003h

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
7 TCF	Transfer Complete Flag Acknowledges a byte transfer; TCF is set 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: • 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

Table continues on the next page...

I2C*x*_S field descriptions (continued)

Field	Description
	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. • 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

41.3.5 I2C Data I/O register (I2C*x*_D)

Address: 4006_6000h base + 4h offset = 4006_6004h

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

I2C*x*_D field descriptions

Field	Description
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.

I2Cx_D field descriptions (continued)

Field	Description
	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. 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).

41.3.6 I2C Control Register 2 (I2Cx_C2)

Address: 4006_6000h base + 5h offset = 4006_6005h

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
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.

41.3.7 I2C Programmable Input Glitch Filter Register (I2Cx_FLT)

Address: 4006_6000h base + 6h offset = 4006_6006h

Bit	7	6	5	4	3	2	1	0
Read	SHEN	STOPF	SSIE	STARTF	FLT			
Write		w1c		w1c				
Reset	0	0	0	0	0	0	0	0

I2Cx_FLT field descriptions

Field	Description
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. 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.

Table continues on the next page...

I2Cx_FLT field descriptions (continued)

Field	Description
	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. **NOTE:** The stop flag is only for the matched slave devices, therefore the master will not respond for 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
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*=1-15d

41.3.8 I2C Range Address register (I2Cx_RA)

Address: 4006_6000h base + 7h offset = 4006_6007h

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

I2Cx_RA field descriptions

Field	Description
7–1 RAD	Range Slave Address

Table continues on the next page...

I2Cx_RA field descriptions (continued)

Field	Description
	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.

41.3.9 I2C SMBus Control and Status register (I2Cx_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: 4006_6000h base + 8h offset = 4006_6008h

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
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.

Table continues on the next page...

I2Cx_SMB field descriptions (continued)

Field	Description
	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. 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

41.3.10 I2C Address Register 2 (I2Cx_A2)

Address: 4006_6000h base + 9h offset = 4006_6009h

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

I2Cx_A2 field descriptions

Field	Description
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.

41.3.11 I2C SCL Low Timeout Register High (I2Cx_SLTH)

Address: 4006_6000h base + Ah offset = 4006_600Ah

Bit	7	6	5	4	3	2	1	0
Read	SSLT[15:8]							
Write								
Reset	0	0	0	0	0	0	0	0

I2Cx_SLTH field descriptions

Field	Description
SSLT[15:8]	SSLT[15:8] Most significant byte of SCL low timeout value that determines the timeout period of SCL low.

41.3.12 I2C SCL Low Timeout Register Low (I2Cx_SLTL)

Address: 4006_6000h base + Bh offset = 4006_600Bh

Bit	7	6	5	4	3	2	1	0
Read	SSLT[7:0]							
Write								
Reset	0	0	0	0	0	0	0	0

I2Cx_SLTL field descriptions

Field	Description
SSLT[7:0]	SSLT[7:0]

I2Cx_SLTL field descriptions (continued)

Field	Description
	Least significant byte of SCL low timeout value that determines the timeout period of SCL low.

41.4 Functional description

This section provides a comprehensive functional description of the I2C module.

41.4.1 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.

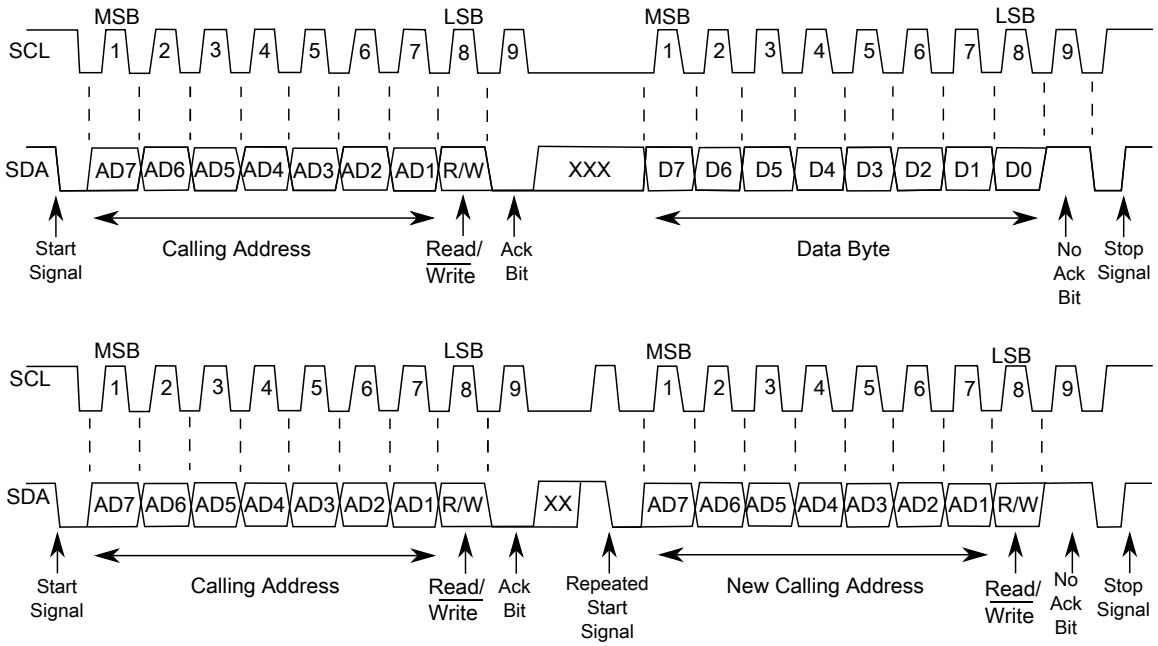


Figure 41-2. I2C bus transmission signals

41.4.1.1 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.

41.4.1.2 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.

41.4.1.3 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.

41.4.1.4 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.

41.4.1.5 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.

41.4.1.6 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.

41.4.1.7 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.

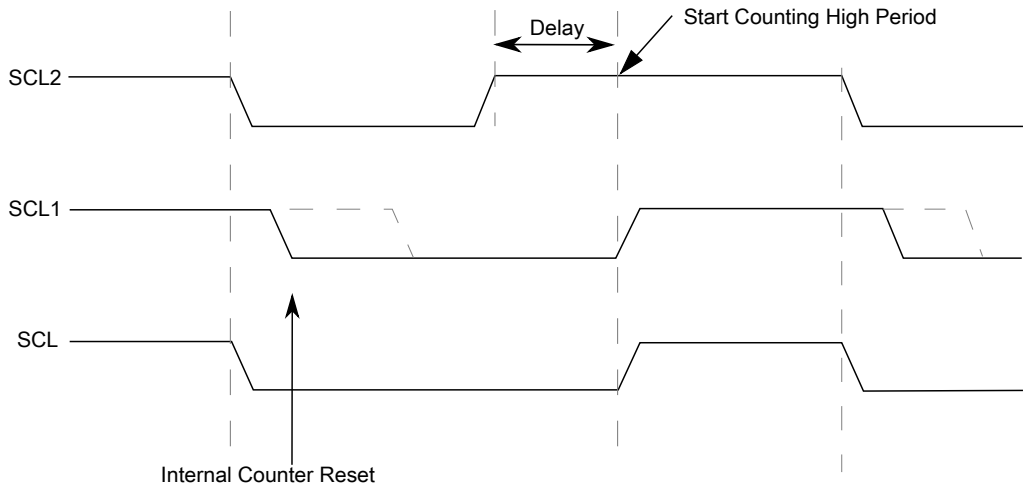


Figure 41-3. I2C clock synchronization

41.4.1.8 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.

41.4.1.9 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.

41.4.1.10 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.

Table 41-2. 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

41.4.2 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.

41.4.2.1 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 41-3. 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.

41.4.2.2 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 41-4. 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.

41.4.3 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.

41.4.4 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.

41.4.4.1 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.

41.4.4.1.1 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.

41.4.4.1.2 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.

41.4.4.1.3 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.

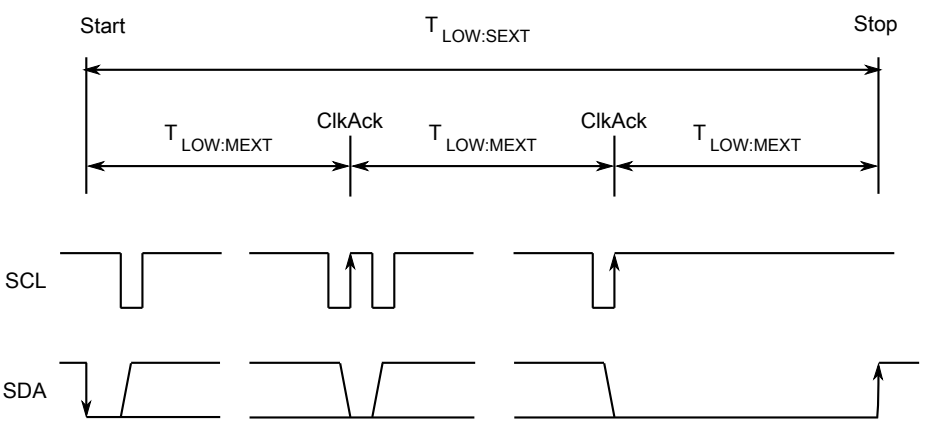


Figure 41-4. Timeout measurement intervals

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.

41.4.4.2 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.

41.4.5 Resets

The I2C module is disabled after a reset. The I2C module cannot cause a core reset.

41.4.6 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 41-5. 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

41.4.6.1 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.

41.4.6.2 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.

41.4.6.3 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 SSIE bits are both set to 1.

41.4.6.4 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.

41.4.6.5 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.

41.4.6.6 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.

41.4.7 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.

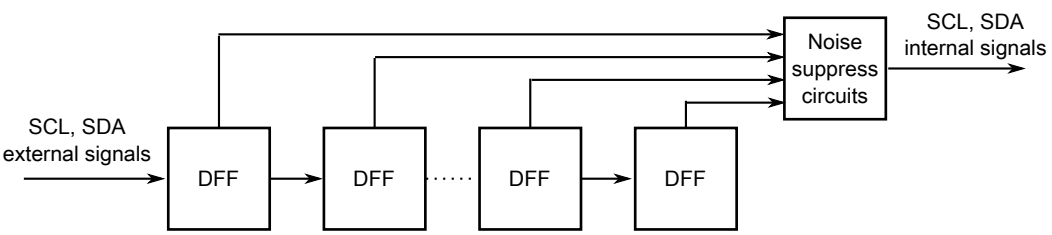


Figure 41-5. Programmable input glitch filter diagram

41.4.8 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).

41.4.9 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.

41.5 Initialization/application information

Module Initialization (Slave)

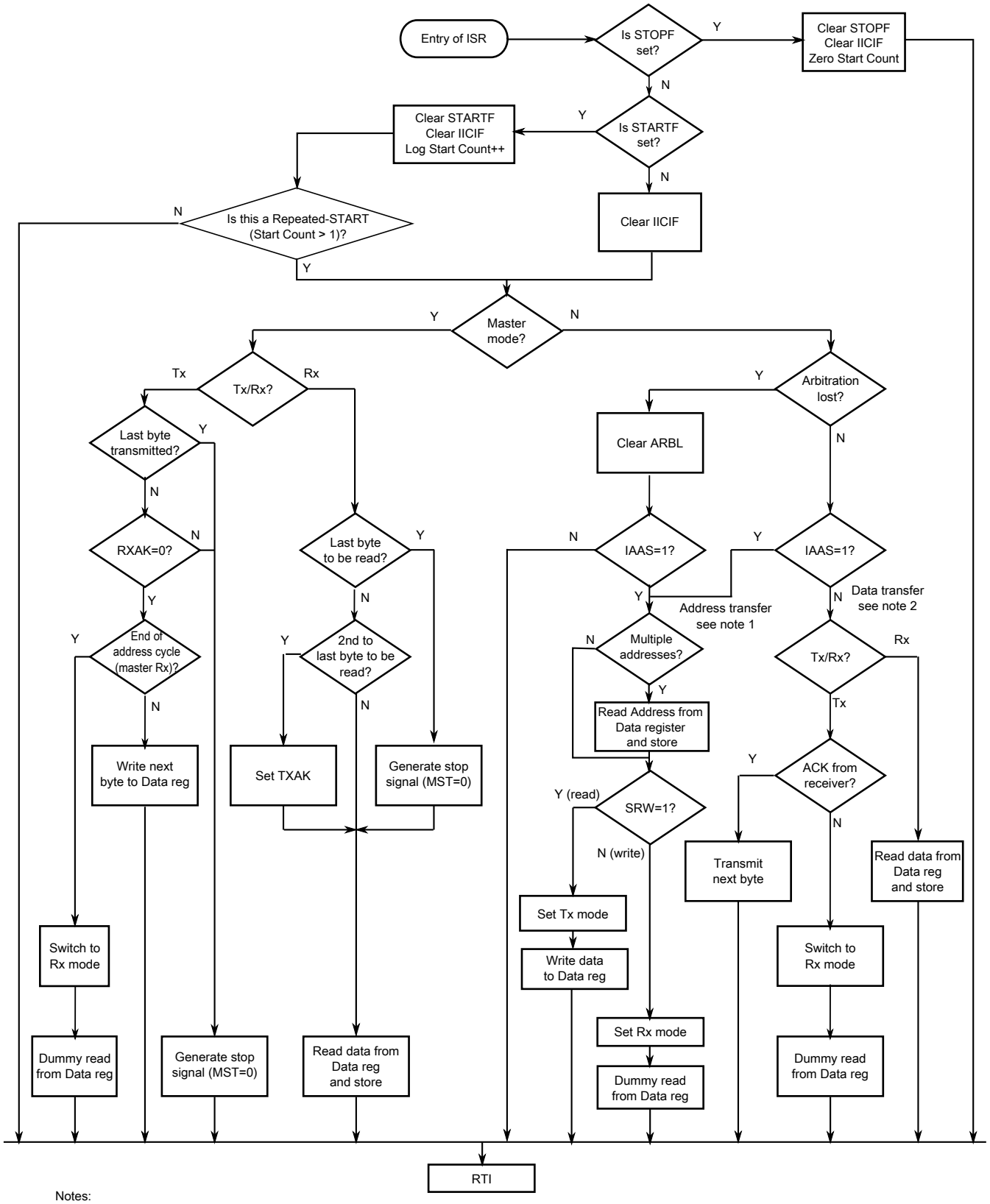
1. Write: Control Register 2
   - to enable or disable general call
   - to select 10-bit or 7-bit addressing mode
2. Write: Address Register 1 to set the slave address

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)

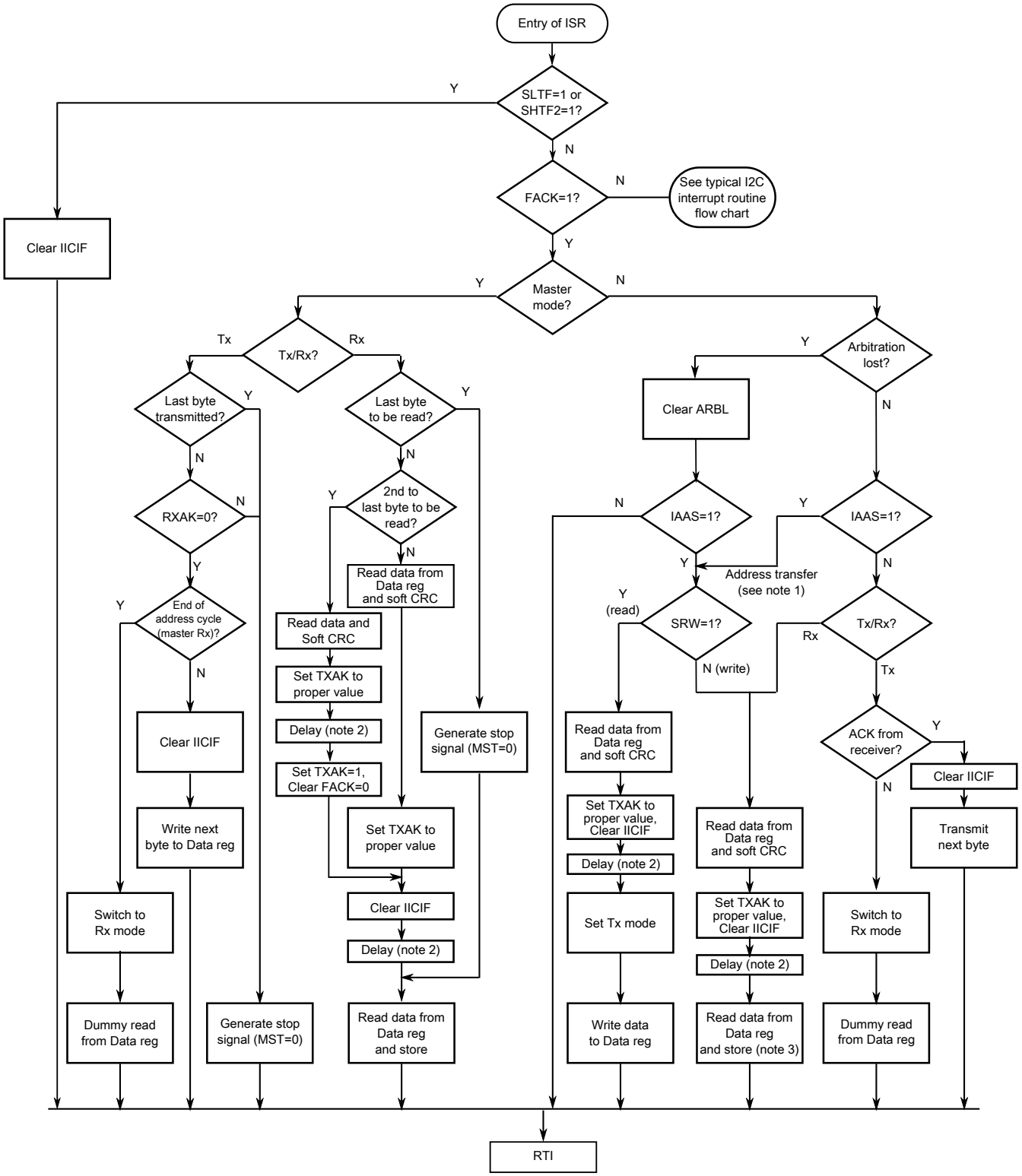
1. Write: Frequency Divider register to set the I2C baud rate (see example in description of ICR)
2. Write: Control Register 1 to enable the I2C module and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown 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 41-6. 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 41-7. Typical I2C SMBus interrupt routine

Chapter 42
Universal Asynchronous Receiver/Transmitter (UART)

42.1 Introduction

NOTE

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

The UART allows asynchronous serial communication with peripheral devices and CPUs.

42.1.1 Features

The UART includes the following features:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- 13-bit baud rate selection with /32 fractional divide, based on the module clock frequency
- Programmable 8-bit or 9-bit data format
- Programmable 1 or 2 stop bits in a data frame.
- Separately enabled transmitter and receiver
- Programmable transmitter output polarity
- Programmable receive input polarity
- Up to 16-bit break character transmission.

- 11-bit break character detection option
- Independent FIFO structure for transmit and receive
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
- Address match feature in the receiver to reduce address mark wakeup ISR overhead
- Ability to select MSB or LSB to be first bit on wire
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Interrupt-driven operation with flags
  - Transmitter data buffer at or below watermark
  - Transmission complete
  - Receiver data buffer at or above watermark
  - Idle receiver input
  - Receiver data buffer overrun
  - Receiver data buffer underflow
  - Transmit data buffer overflow
  - Noise error
  - Framing error
  - Parity error
  - Active edge on receive pin
  - LIN break detect
- Receiver framing error detection
- Hardware parity generation and checking
- 1/16 bit-time noise detection
- DMA interface

42.1.2 Modes of operation

The UART functions in the same way in all the normal modes.

It has the following low power modes:

- Wait mode
- Stop mode

42.1.2.1 Run mode

This is the normal mode of operation.

42.1.2.2 Wait mode

UART operation in the Wait mode depends on the state of the C1[UARTSWAI] field.

- If C1[UARTSWAI] is cleared, and the CPU is in Wait mode, the UART operates normally.
- If C1[UARTSWAI] is set, and the CPU is in Wait mode, the UART clock generation ceases and the UART module enters a power conservation state.

Setting C1[UARTSWAI] does not affect the state of the C2[RE] or C2[TE].

If C1[UARTSWAI] is set, any ongoing transmission or reception stops at the Wait mode entry. The transmission or reception resumes when either an internal or external interrupt brings the CPU out of Wait mode. Bringing the CPU out of Wait mode by reset aborts any ongoing transmission or reception and resets the UART.

42.1.2.3 Stop mode

The UART is inactive during Stop mode for reduced power consumption. The STOP instruction does not affect the UART register states, but the UART module clock is disabled. The UART operation resumes after an external interrupt brings the CPU out of Stop mode. Bringing the CPU out of Stop mode by reset aborts any ongoing transmission or reception and resets the UART.

42.2 UART signal descriptions

The UART signals are shown in the following table.

Table 42-1. UART signal descriptions

Signal	Description	I/O
$\overline{\text{CTS}}$	Clear to send	I
$\overline{\text{RTS}}$	Request to send	O
RXD	Receive data	I
TXD	Transmit data	O

42.2.1 Detailed signal descriptions

The detailed signal descriptions of the UART are shown in the following table.

Table 42-2. UART—Detailed signal descriptions

Signal	I/O	Description	
$\overline{\text{CTS}}$	I	Clear to send. Indicates whether the UART can start transmitting data when flow control is enabled.	
		State meaning	Asserted—Data transmission can start. Negated—Data transmission cannot start.
		Timing	Assertion—When transmitting device's $\overline{\text{RTS}}$ asserts. Negation—When transmitting device's $\overline{\text{RTS}}$ deasserts.
$\overline{\text{RTS}}$	O	Request to send. When driven by the receiver, indicates whether the UART is ready to receive data. When driven by the transmitter, can enable an external transceiver during transmission.	
		State meaning	Asserted—When driven by the receiver, ready to receive data. When driven by the transmitter, enable the external transmitter. Negated—When driven by the receiver, not ready to receive data. When driven by the transmitter, disable the external transmitter.
		Timing	Assertion—Can occur at any time; can assert asynchronously to the other input signals. Negation—Can occur at any time; can deassert asynchronously to the other input signals.
RXD	I	Receive data. Serial data input to receiver.	
		State meaning	Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.
		Timing	Sampled at a frequency determined by the module clock divided by the baud rate.
TXD	O	Transmit data. Serial data output from transmitter.	

Table continues on the next page...

Table 42-2. UART—Detailed signal descriptions (continued)

Signal	I/O	Description	
		State meaning	Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.
		Timing	Driven at the beginning or within a bit time according to the bit encoding method along with other configuration settings. Otherwise, transmissions are independent of reception timing.

42.3 Memory map and registers

This section provides a detailed description of all memory and registers.

Accessing reserved addresses within the memory map results in a transfer error. None of the contents of the implemented addresses are modified as a result of that access.

Only byte accesses are supported.

UART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_A000	UART Baud Rate Registers: High (UART0_BDH)	8	R/W	00h	42.3.1/1091
4006_A001	UART Baud Rate Registers: Low (UART0_BDL)	8	R/W	04h	42.3.2/1092
4006_A002	UART Control Register 1 (UART0_C1)	8	R/W	00h	42.3.3/1092
4006_A003	UART Control Register 2 (UART0_C2)	8	R/W	00h	42.3.4/1094
4006_A004	UART Status Register 1 (UART0_S1)	8	R	C0h	42.3.5/1096
4006_A005	UART Status Register 2 (UART0_S2)	8	R/W	00h	42.3.6/1099
4006_A006	UART Control Register 3 (UART0_C3)	8	R/W	00h	42.3.7/1100
4006_A007	UART Data Register (UART0_D)	8	R/W	00h	42.3.8/1102
4006_A008	UART Match Address Registers 1 (UART0_MA1)	8	R/W	00h	42.3.9/1103
4006_A009	UART Match Address Registers 2 (UART0_MA2)	8	R/W	00h	42.3.10/ 1103
4006_A00A	UART Control Register 4 (UART0_C4)	8	R/W	00h	42.3.11/ 1103
4006_A00B	UART Control Register 5 (UART0_C5)	8	R/W	00h	42.3.12/ 1104
4006_A00C	UART Extended Data Register (UART0_ED)	8	R	00h	42.3.13/ 1105
4006_A00D	UART Modem Register (UART0_MODEM)	8	R/W	00h	42.3.14/ 1106
4006_A010	UART FIFO Parameters (UART0_PFIFO)	8	R/W	See section	42.3.15/ 1107

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_A011	UART FIFO Control Register (UART0_CFIFO)	8	R/W	00h	42.3.16/ 1109
4006_A012	UART FIFO Status Register (UART0_SFIFO)	8	R/W	C0h	42.3.17/ 1110
4006_A013	UART FIFO Transmit Watermark (UART0_TWFIFO)	8	R/W	00h	42.3.18/ 1111
4006_A014	UART FIFO Transmit Count (UART0_TCFIFO)	8	R	00h	42.3.19/ 1112
4006_A015	UART FIFO Receive Watermark (UART0_RWFIFO)	8	R/W	01h	42.3.20/ 1112
4006_A016	UART FIFO Receive Count (UART0_RCFIFO)	8	R	00h	42.3.21/ 1113
4006_B000	UART Baud Rate Registers: High (UART1_BDH)	8	R/W	00h	42.3.1/1091
4006_B001	UART Baud Rate Registers: Low (UART1_BDL)	8	R/W	04h	42.3.2/1092
4006_B002	UART Control Register 1 (UART1_C1)	8	R/W	00h	42.3.3/1092
4006_B003	UART Control Register 2 (UART1_C2)	8	R/W	00h	42.3.4/1094
4006_B004	UART Status Register 1 (UART1_S1)	8	R	C0h	42.3.5/1096
4006_B005	UART Status Register 2 (UART1_S2)	8	R/W	00h	42.3.6/1099
4006_B006	UART Control Register 3 (UART1_C3)	8	R/W	00h	42.3.7/1100
4006_B007	UART Data Register (UART1_D)	8	R/W	00h	42.3.8/1102
4006_B008	UART Match Address Registers 1 (UART1_MA1)	8	R/W	00h	42.3.9/1103
4006_B009	UART Match Address Registers 2 (UART1_MA2)	8	R/W	00h	42.3.10/ 1103
4006_B00A	UART Control Register 4 (UART1_C4)	8	R/W	00h	42.3.11/ 1103
4006_B00B	UART Control Register 5 (UART1_C5)	8	R/W	00h	42.3.12/ 1104
4006_B00C	UART Extended Data Register (UART1_ED)	8	R	00h	42.3.13/ 1105
4006_B00D	UART Modem Register (UART1_MODEM)	8	R/W	00h	42.3.14/ 1106
4006_B010	UART FIFO Parameters (UART1_PFIFO)	8	R/W	See section	42.3.15/ 1107
4006_B011	UART FIFO Control Register (UART1_CFIFO)	8	R/W	00h	42.3.16/ 1109
4006_B012	UART FIFO Status Register (UART1_SFIFO)	8	R/W	C0h	42.3.17/ 1110
4006_B013	UART FIFO Transmit Watermark (UART1_TWFIFO)	8	R/W	00h	42.3.18/ 1111
4006_B014	UART FIFO Transmit Count (UART1_TCFIFO)	8	R	00h	42.3.19/ 1112
4006_B015	UART FIFO Receive Watermark (UART1_RWFIFO)	8	R/W	01h	42.3.20/ 1112

Table continues on the next page...

UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_B016	UART FIFO Receive Count (UART1_RCFIFO)	8	R	00h	42.3.21/ 1113

42.3.1 UART Baud Rate Registers: High (UARTx_BDH)

This register, along with the BDL register, controls the prescale divisor for UART baud rate generation. To update the 13-bit baud rate setting (SBR[12:0]), first write to BDH to buffer the high half of the new value and then write to BDL. The working value in BDH does not change until BDL is written.

BDL is reset to a nonzero value, but after reset, the baud rate generator remains disabled until the first time the receiver or transmitter is enabled, that is, when C2[RE] or C2[TE] is set.

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read Write	LBKDIE	RXEDGIE	SBNS	SBR				
Reset	0	0	0	0	0	0	0	0

UARTx_BDH field descriptions

Field	Description
7 LBKDIE	LIN Break Detect Interrupt Enable Enables the LIN break detect flag, LBKDIF, to generate interrupt requests, 0 LBKDIF interrupt requests disabled. 1 LBKDIF interrupt requests enabled.
6 RXEDGIE	RxD Input Active Edge Interrupt Enable Enables the receive input active edge, RXEDGIF, to generate interrupt requests. 0 Hardware interrupts from RXEDGIF disabled using polling. 1 RXEDGIF interrupt request enabled.
5 SBNS	Stop Bit Number Select SBNS selects the number of stop bits present in a data frame. This field valid for all 8, 9 and 10 bit data formats available. 0 Data frame consists of a single stop bit. 1 Data frame consists of two stop bits.
SBR	UART Baud Rate Bits The baud rate for the UART is determined by the 13 SBR fields. See Baud rate generation for details.

Table continues on the next page...

UARTx_BDH field descriptions (continued)

Field	Description
	NOTE: • The baud rate generator is disabled until C2[TE] or C2[RE] is set for the first time after reset.The baud rate generator is disabled when SBR = 0. • Writing to BDH has no effect without writing to BDL, because writing to BDH puts the data in a temporary location until BDL is written.

42.3.2 UART Baud Rate Registers: Low (UARTx_BDL)

This register, along with the BDH register, controls the prescale divisor for UART baud rate generation. To update the 13-bit baud rate setting, SBR[12:0], first write to BDH to buffer the high half of the new value and then write to BDL. The working value in BDH does not change until BDL is written. BDL is reset to a nonzero value, but after reset, the baud rate generator remains disabled until the first time the receiver or transmitter is enabled, that is, when C2[RE] or C2[TE] is set.

Address: Base address + 1h offset

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

UARTx_BDL field descriptions

Field	Description
SBR	UART Baud Rate Bits The baud rate for the UART is determined by the 13 SBR fields. See Baud rate generation for details. **NOTE:** • The baud rate generator is disabled until C2[TE] or C2[RE] is set for the first time after reset.The baud rate generator is disabled when SBR = 0. • Writing to BDH has no effect without writing to BDL, because writing to BDH puts the data in a temporary location until BDL is written.

42.3.3 UART Control Register 1 (UARTx_C1)

This read/write register controls various optional features of the UART system.

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read Write	LOOPS	UARTSWAI	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

UARTx_C1 field descriptions

Field	Description
7 LOOPS	Loop Mode Select When LOOPS is set, the RxD pin is disconnected from the UART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function. 0 Normal operation. 1 Loop mode where transmitter output is internally connected to receiver input. The receiver input is determined by RSRC.
6 UARTSWAI	UART Stops in Wait Mode 0 UART clock continues to run in Wait mode. 1 UART clock freezes while CPU is in Wait mode.
5 RSRC	Receiver Source Select This field has no meaning or effect unless the LOOPS field is set. When LOOPS is set, the RSRC field determines the source for the receiver shift register input. 0 Selects internal loop back mode. The receiver input is internally connected to transmitter output. 1 Single wire UART mode where the receiver input is connected to the transmit pin input signal.
4 M	9-bit or 8-bit Mode Select 0 Normal—start + 8 data bits (MSB/LSB first as determined by MSBF) + stop. 1 Use—start + 9 data bits (MSB/LSB first as determined by MSBF) + stop.
3 WAKE	Receiver Wakeup Method Select Determines which condition wakes the UART: • Address mark in the most significant bit position of a received data character, or • An idle condition on the receive pin input signal. 0 Idle line wakeup. 1 Address mark wakeup.
2 ILT	Idle Line Type Select Determines when the receiver starts counting logic 1s as idle character bits. The count begins either after a valid start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. **NOTE:** • In case the UART is programmed with ILT = 1, a logic of 1'b0 is automatically shifted after a received stop bit, therefore resetting the idle count. • In case the UART is programmed for IDLE line wakeup (RWU = 1 and WAKE = 0), ILT has no effect on when the receiver starts counting logic 1s as idle character bits. In idle line wakeup, an idle character is recognized at anytime the receiver sees 10, 11, or 12 1s depending on the M, PE, and C4[M10] fields. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.
1 PE	Parity Enable Enables the parity function. When parity is enabled, parity function inserts a parity bit in the bit position immediately preceding the stop bit.

Table continues on the next page...

UARTx_C1 field descriptions (continued)

Field	Description
	0 Parity function disabled. 1 Parity function enabled.
0 PT	Parity Type Determines whether the UART generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit. 0 Even parity. 1 Odd parity.

42.3.4 UART Control Register 2 (UARTx_C2)

This register can be read or written at any time.

Address: Base address + 3h offset

Bit	7	6	5	4	3	2	1	0
Read Write	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

UARTx_C2 field descriptions

Field	Description
7 TIE	Transmitter Interrupt or DMA Transfer Enable. Enables S1[TDRE] to generate interrupt requests or DMA transfer requests, based on the state of C5[TDMAS]. **NOTE:** If C2[TIE] and C5[TDMAS] are both set, then TCIE must be cleared, and D[D] must not be written unless servicing a DMA request. 0 TDRE interrupt and DMA transfer requests disabled. 1 TDRE interrupt or DMA transfer requests enabled.
6 TCIE	Transmission Complete Interrupt Enable Enables the transmission complete flag, S1[TC], to generate interrupt requests . 0 TC interrupt requests disabled. 1 TC interrupt requests enabled.
5 RIE	Receiver Full Interrupt or DMA Transfer Enable Enables S1[RDRF] to generate interrupt requests or DMA transfer requests, based on the state of C5[RDMAS]. 0 RDRF interrupt and DMA transfer requests disabled. 1 RDRF interrupt or DMA transfer requests enabled.
4 ILIE	Idle Line Interrupt Enable

Table continues on the next page...

UARTx_C2 field descriptions (continued)

Field	Description
	Enables the idle line flag, S1[IDLE], to generate interrupt requests 0 IDLE interrupt requests disabled. 1 IDLE interrupt requests enabled.
3 TE	Transmitter Enable Enables the UART transmitter. TE can be used to queue an idle preamble by clearing and then setting TE. 0 Transmitter off. 1 Transmitter on.
2 RE	Receiver Enable Enables the UART receiver. 0 Receiver off. 1 Receiver on.
1 RWU	Receiver Wakeup Control This field can be set to place the UART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when C1[WAKE] is clear or an address match when C1[WAKE] is set. **NOTE:** RWU must be set only with C1[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by S2[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the UART will discard data. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to reasserted. 0 Normal operation. 1 RWU enables the wakeup function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU.
0 SBK	Send Break Toggling SBK sends one break character from the following: See Transmitting break characters for the number of logic 0s for the different configurations. Toggling implies clearing the SBK field before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send complete break characters (10, 11, or 12 bits, or 13 or 14 bits). Ensure that C2[TE] is asserted atleast 1 clock before assertion of this bit. • 10, 11, or 12 logic 0s if S2[BRK13] is cleared • 13 or 14 logic 0s if S2[BRK13] is set. 0 Normal transmitter operation. 1 Queue break characters to be sent.

42.3.5 UART Status Register 1 (UARTx_S1)

The S1 register provides inputs to the MCU for generation of UART interrupts or DMA requests. This register can also be polled by the MCU to check the status of its fields. To clear a flag, the status register should be read followed by a read or write to D register, depending on the interrupt flag type. Other instructions can be executed between the two steps as long the handling of I/O is not compromised, but the order of operations is important for flag clearing. When a flag is configured to trigger a DMA request, assertion of the associated DMA done signal from the DMA controller clears the flag.

NOTE

- If the condition that results in the assertion of the flag, interrupt, or DMA request is not resolved prior to clearing the flag, the flag, and interrupt/DMA request, reasserts. For example, if the DMA or interrupt service routine fails to write sufficient data to the transmit buffer to raise it above the watermark level, the flag reasserts and generates another interrupt or DMA request.
- Reading an empty data register to clear one of the flags of the S1 register causes the FIFO pointers to become misaligned. A receive FIFO flush reinitializes the pointers. A better way to prevent this situation is to always leave one byte in FIFO and this byte will be read eventually in clearing the flag bit.

Address: Base address + 4h offset

Bit	7	6	5	4	3	2	1	0
Read	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write								
Reset	1	1	0	0	0	0	0	0

UARTx_S1 field descriptions

Field	Description
7 TDRE	Transmit Data Register Empty Flag TDRE will set when the number of datawords in the transmit buffer (D and C3[T8])is equal to or less than the number indicated by TWFIFO[TXWATER]. A character that is in the process of being transmitted is not included in the count. To clear TDRE, read S1 when TDRE is set and then write to the UART data register (D). For more efficient interrupt servicing, all data except the final value to be written to the buffer must be written to D/C3[T8]. Then S1 can be read before writing the final data value, resulting in the clearing of the TRDE flag. This is more efficient because the TDRE reasserts until the watermark has been exceeded. So, attempting to clear the TDRE with every write will be ineffective until sufficient data has been written.

Table continues on the next page...

UARTx_S1 field descriptions (continued)

Field	Description
	0 The amount of data in the transmit buffer is greater than the value indicated by TWFIFO[TXWATER]. 1 The amount of data in the transmit buffer is less than or equal to the value indicated by TWFIFO[TXWATER] at some point in time since the flag has been cleared.
6 TC	Transmit Complete Flag TC is set when the transmit buffer is empty and no data, preamble, or break character is being transmitted. When TC is set, the transmit data output signal becomes idle (logic 1). TC is cleared by reading S1 with TC set and then doing one of the following: • Writing to D to transmit new data. • Queuing a preamble by clearing and then setting C2[TE]. • Queuing a break character by writing 1 to SBK in C2. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete).
5 RDRF	Receive Data Register Full Flag RDRF is set when the number of datawords in the receive buffer is equal to or more than the number indicated by RWFIFO[RXWATER]. A dataword that is in the process of being received is not included in the count. To clear RDRF, read S1 when RDRF is set and then read D. For more efficient interrupt and DMA operation, read all data except the final value from the buffer, using D/C3[T8]/ED. Then read S1 and the final data value, resulting in the clearing of the RDRF flag. Even if RDRF is set, data will continue to be received until an overrun condition occurs.RDRF is prevented from setting while S2[LBKDE] is set. Additionally, when S2[LBKDE] is set, the received datawords are stored in the receive buffer but over-write each other. 0 The number of datawords in the receive buffer is less than the number indicated by RXWATER. 1 The number of datawords in the receive buffer is equal to or greater than the number indicated by RXWATER at some point in time since this flag was last cleared.
4 IDLE	Idle Line Flag After the IDLE flag is cleared, a frame must be received (although not necessarily stored in the data buffer, for example if C2[RWU] is set), or a LIN break character must set the S2[LBKDIF] flag before an idle condition can set the IDLE flag. To clear IDLE, read UART status S1 with IDLE set and then read D. IDLE is set when either of the following appear on the receiver input: • 10 consecutive logic 1s if C1[M] = 0 • 11 consecutive logic 1s if C1[M] = 1 and C4[M10] = 0 • 12 consecutive logic 1s if C1[M] = 1, C4[M10] = 1, and C1[PE] = 1 **NOTE:** When RWU is set and WAKE is cleared, an idle line condition sets the IDLE flag if RWUID is set, else the IDLE flag does not become set. 0 Receiver input is either active now or has never become active since the IDLE flag was last cleared. 1 Receiver input has become idle or the flag has not been cleared since it last asserted.
3 OR	Receiver Overrun Flag OR is set when software fails to prevent the receive data register from overflowing with data. The OR bit is set immediately after the stop bit has been completely received for the dataword that overflows the buffer and all the other error flags (FE, NF, and PF) are prevented from setting. The data in the shift register is lost, but the data already in the UART data registers is not affected. If the OR flag is set, no data is stored in the data buffer even if sufficient room exists. Additionally, while the OR flag is set, the RDRF and IDLE flags are blocked from asserting, that is, transition from an inactive to an active state. To clear OR, read S1 when OR is set and then read D. See functional description for more details regarding the operation of the OR bit.If LBKDE is enabled and a LIN Break is detected, the OR field asserts if S2[LBKDIF] is not cleared before the next data character is received.

Table continues on the next page...

UARTx_S1 field descriptions (continued)

Field	Description
	0 No overrun has occurred since the last time the flag was cleared. 1 Overrun has occurred or the overrun flag has not been cleared since the last overrun occured.
2 NF	Noise Flag NF is set when the UART detects noise on the receiver input. NF does not become set in the case of an overrun or while the LIN break detect feature is enabled (S2[LBKDE] = 1). When NF is set, it indicates only that a dataword has been received with noise since the last time it was cleared. There is no guarantee that the first dataword read from the receive buffer has noise or that there is only one dataword in the buffer that was received with noise unless the receive buffer has a depth of one. To clear NF, read S1 and then read D. 0 No noise detected since the last time this flag was cleared. If the receive buffer has a depth greater than 1 then there may be data in the receiver buffer that was received with noise. 1 At least one dataword was received with noise detected since the last time the flag was cleared.
1 FE	Framing Error Flag FE is set when a logic 0 is accepted as the stop bit. When BDH[SBNS] is set, then FE will set when a logic 0 is accepted for either of the two stop bits. FE does not set in the case of an overrun or while the LIN break detect feature is enabled (S2[LBKDE] = 1). FE inhibits further data reception until it is cleared. To clear FE, read S1 with FE set and then read D. The last data in the receive buffer represents the data that was received with the frame error enabled. 0 No framing error detected. 1 Framing error.
0 PF	Parity Error Flag PF is set when PE is set and the parity of the received data does not match its parity bit. The PF is not set in the case of an overrun condition. When PF is set, it indicates only that a dataword was received with parity error since the last time it was cleared. There is no guarantee that the first dataword read from the receive buffer has a parity error or that there is only one dataword in the buffer that was received with a parity error, unless the receive buffer has a depth of one. To clear PF, read S1 and then read D., S2[LBKDE] is disabled,Within the receive buffer structure the received dataword is tagged if it is received with a parity error. This information is available by reading the ED register prior to reading the D register. 0 No parity error detected since the last time this flag was cleared. If the receive buffer has a depth greater than 1, then there may be data in the receive buffer what was received with a parity error. 1 At least one dataword was received with a parity error since the last time this flag was cleared.

42.3.6 UART Status Register 2 (UARTx_S2)

The S2 register provides inputs to the MCU for generation of UART interrupts or DMA requests. Also, this register can be polled by the MCU to check the status of these bits. This register can be read or written at any time, with the exception of the MSBF and RXINV bits, which should be changed by the user only between transmit and receive packets.

Address: Base address + 5h offset

Bit	7	6	5	4	3	2	1	0
Read	LBKDIF	RXEDGIF	MSBF	RXINV	RWUID	BRK13	LBKDE	RAF
Write	w1c	w1c						
Reset	0	0	0	0	0	0	0	0

UARTx_S2 field descriptions

Field	Description
7 LBKDIF	LIN Break Detect Interrupt Flag LBKDIF is set when LBKDE is set and a LIN break character is detected on the receiver input. The LIN break characters are 11 consecutive logic 0s if C1[M] = 0 or 12 consecutive logic 0s if C1[M] = 1. LBKDIF is set after receiving the last LIN break character. LBKDIF is cleared by writing a 1 to it. 0 No LIN break character detected. 1 LIN break character detected.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag RXEDGIF is set when an active edge occurs on the RxD pin. The active edge is falling if RXINV = 0, and rising if RXINV=1. RXEDGIF is cleared by writing a 1 to it. See for additional details. RXEDGIF description **NOTE:** The active edge is detected only in two wire mode and on receiving data coming from the RxD pin. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
5 MSBF	Most Significant Bit First Setting this field reverses the order of the bits that are transmitted and received on the wire. This field does not affect the polarity of the bits, the location of the parity bit, or the location of the start or stop bits. 0 LSB (bit0) is the first bit that is transmitted following the start bit. Further, the first bit received after the start bit is identified as bit0. 1 MSB (bit8, bit7 or bit6) is the first bit that is transmitted following the start bit, depending on the setting of C1[M] and C1[PE]. Further, the first bit received after the start bit is identified as bit8, bit7, or bit6, depending on the setting of C1[M] and C1[PE].
4 RXINV	Receive Data Inversion Setting this field reverses the polarity of the received data input. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. **NOTE:** Setting RXINV inverts the RxD input for data bits, start and stop bits, break, and idle.

Table continues on the next page...

UARTx_S2 field descriptions (continued)

Field	Description
	0 Receive data is not inverted. 1 Receive data is inverted.
3 RWUID	Receive Wakeup Idle Detect When RWU is set and WAKE is cleared, this field controls whether the idle character that wakes the receiver sets S1[IDLE]. 0 S1[IDLE] is not set upon detection of an idle character. 1 S1[IDLE] is set upon detection of an idle character.
2 BRK13	Break Transmit Character Length Determines whether the transmit break character is 10, 11, or 12 bits long, or 13 or 14 bits long. See for the length of the break character for the different configurations. The detection of a framing error is not affected by this field. Transmitting break characters 0 Break character is 10, 11, or 12 bits long. 1 Break character is 13 or 14 bits long.
1 LBKDE	LIN Break Detection Enable Enables the LIN Break detection feature. While LBKDE is set, S1[RDRF], S1[NF], S1[FE], and S1[PF] are prevented from setting. When LBKDE is set, see . Overrun operation 0 Break character detection is disabled. 1 Break character is detected at length of 11 bit times if C1[M] = 0 or 12 bits time if C1[M] = 1.
0 RAF	Receiver Active Flag RAF is set when the UART receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character. 0 UART receiver idle/inactive waiting for a start bit. 1 UART receiver active, RxD input not idle.

42.3.7 UART Control Register 3 (UARTx_C3)

Writing R8 does not have any effect. TXDIR and TXINV can be changed only between transmit and receive packets.

Address: Base address + 6h offset

Bit	7	6	5	4	3	2	1	0
Read	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_C3 field descriptions

Field	Description
7 R8	Received Bit 8

Table continues on the next page...

UARTx_C3 field descriptions (continued)

Field	Description
	R8 is the ninth data bit received when the UART is configured for 9-bit data format, that is, if C1[M] = 1 or C4[M10] = 1. The R8 value corresponds to the current data value in the UARTx_D register. To read the 9th bit, read the value of UARTx_C3[R8], then read the UARTx_D register.
6 T8	Transmit Bit 8 T8 is the ninth data bit transmitted when the UART is configured for 9-bit data format, that is, if C1[M] = 1 or C4[M10] = 1. **NOTE:** If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten. The same value is transmitted until T8 is rewritten. To correctly transmit the 9th bit, write UARTx_C3[T8] to the desired value, then write the UARTx_D register with the remaining data.
5 TXDIR	Transmitter Pin Data Direction in Single-Wire mode Determines whether the TXD pin is used as an input or output in the single-wire mode of operation. This field is relevant only to the single wire mode. 0 TXD pin is an input in single wire mode. 1 TXD pin is an output in single wire mode.
4 TXINV	Transmit Data Inversion. Setting this field reverses the polarity of the transmitted data output. In NRZ format, a one is represented by a mark and a zero is represented by a space for normal polarity, and the opposite for inverted polarity. **NOTE:** Setting TXINV inverts all transmitted values, including idle, break, start, and stop bits. In loop mode, if TXINV is set, the receiver gets the transmit inversion bit when RXINV is disabled. 0 Transmit data is not inverted. 1 Transmit data is inverted.
3 ORIE	Overrun Error Interrupt Enable Enables the overrun error flag, S1[OR], to generate interrupt requests. 0 OR interrupts are disabled. 1 OR interrupt requests are enabled.
2 NEIE	Noise Error Interrupt Enable Enables the noise flag, S1[NF], to generate interrupt requests. 0 NF interrupt requests are disabled. 1 NF interrupt requests are enabled.
1 FEIE	Framing Error Interrupt Enable Enables the framing error flag, S1[FE], to generate interrupt requests. 0 FE interrupt requests are disabled. 1 FE interrupt requests are enabled.
0 PEIE	Parity Error Interrupt Enable Enables the parity error flag, S1[PF], to generate interrupt requests. 0 PF interrupt requests are disabled. 1 PF interrupt requests are enabled.

42.3.8 UART Data Register (UARTx_D)

This register is actually two separate registers. Reads return the contents of the read-only receive data register and writes go to the write-only transmit data register.

NOTE

- In 8-bit or 9-bit data format, only UART data register (D) needs to be accessed to clear the S1[RDRF] bit (assuming receiver buffer level is less than RWFIFO[RXWATER]). The C3 register needs to be read, prior to the D register, only if the ninth bit of data needs to be captured. Similarly, the ED register needs to be read, prior to the D register, only if the additional flag data for the dataword needs to be captured.
- In the normal 8-bit mode (M bit cleared) if the parity is enabled, you get seven data bits and one parity bit. That one parity bit is loaded into the D register. So, for the data bits, mask off the parity bit from the value you read out of this register.
- When transmitting in 9-bit data format and using 8-bit write instructions, write first to transmit bit 8 in UART control register 3 (C3[T8]), then D. A write to C3[T8] stores the data in a temporary register. If D register is written first, and then the new data on data bus is stored in D, the temporary value written by the last write to C3[T8] gets stored in the C3[T8] register.

Address: Base address + 7h offset

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

UARTx_D field descriptions

Field	Description
RT	Reads return the contents of the read-only receive data register and writes go to the write-only transmit data register.

42.3.9 UART Match Address Registers 1 (UARTx_MA1)

The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated C4[MAEN] field is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. These registers can be read and written at anytime.

Address: Base address + 8h offset

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

UARTx_MA1 field descriptions

Field	Description
MA	Match Address

42.3.10 UART Match Address Registers 2 (UARTx_MA2)

These registers can be read and written at anytime. The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated C4[MAEN] field is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded.

Address: Base address + 9h offset

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

UARTx_MA2 field descriptions

Field	Description
MA	Match Address

42.3.11 UART Control Register 4 (UARTx_C4)

Address: Base address + Ah offset

Bit	7	6	5	4	3	2	1	0
Read Write	MAEN1	MAEN2	M10	BRFA				
Reset	0	0	0	0	0	0	0	0

UARTx_C4 field descriptions

Field	Description
7 MAEN1	Match Address Mode Enable 1 See Match address operation for more information. 0 All data received is transferred to the data buffer if MAEN2 is cleared. 1 All data received with the most significant bit cleared, is discarded. All data received with the most significant bit set, is compared with contents of MA1 register. If no match occurs, the data is discarded. If match occurs, data is transferred to the data buffer.
6 MAEN2	Match Address Mode Enable 2 See Match address operation for more information. 0 All data received is transferred to the data buffer if MAEN1 is cleared. 1 All data received with the most significant bit cleared, is discarded. All data received with the most significant bit set, is compared with contents of MA2 register. If no match occurs, the data is discarded. If a match occurs, data is transferred to the data buffer.
5 M10	10-bit Mode select Causes a tenth, non-memory mapped bit to be part of the serial transmission. This tenth bit is generated and interpreted as a parity bit. The M10 field does not affect the LIN send or detect break behavior. If M10 is set, then both C1[M] and C1[PE] must also be set. See Data format for more information. 0 The parity bit is the ninth bit in the serial transmission. 1 The parity bit is the tenth bit in the serial transmission.
BRFA	Baud Rate Fine Adjust This bit field is used to add more timing resolution to the average baud frequency, in increments of 1/32. See Baud rate generation for more information.

42.3.12 UART Control Register 5 (UARTx_C5)

Address: Base address + Bh offset

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

UARTx_C5 field descriptions

Field	Description
7 TDMAS	Transmitter DMA Select Configures the transmit data register empty flag, S1[TDRE], to generate interrupt or DMA requests if C2[TIE] is set. **NOTE:** • If C2[TIE] is cleared, TDRE DMA and TDRE interrupt request signals are not asserted when the TDRE flag is set, regardless of the state of TDMAS. • If C2[TIE] and TDMAS are both set, then C2[TCIE] must be cleared, and D must not be written unless a DMA request is being serviced.

Table continues on the next page...

UARTx_C5 field descriptions (continued)

Field	Description
	0 If C2[TIE] is set and the S1[TDRE] flag is set, the TDRE interrupt request signal is asserted to request interrupt service. 1 If C2[TIE] is set and the S1[TDRE] flag is set, the TDRE DMA request signal is asserted to request a DMA transfer.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 RDMAS	Receiver Full DMA Select Configures the receiver data register full flag, S1[RDRF], to generate interrupt or DMA requests if C2[RIE] is set. **NOTE:** If C2[RIE] is cleared, and S1[RDRF] is set, the RDRF DMA and RDFR interrupt request signals are not asserted, regardless of the state of RDMAS. 0 If C2[RIE] and S1[RDRF] are set, the RDFR interrupt request signal is asserted to request an interrupt service. 1 If C2[RIE] and S1[RDRF] are set, the RDRF DMA request signal is asserted to request a DMA transfer.
4 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.

42.3.13 UART Extended Data Register (UARTx_ED)

This register contains additional information flags that are stored with a received dataword. This register may be read at any time but contains valid data only if there is a dataword in the receive FIFO.

NOTE

- The data contained in this register represents additional information regarding the conditions on which a dataword was received. The importance of this data varies with the application, and in some cases maybe completely optional. These fields automatically update to reflect the conditions of the next dataword whenever D is read.
- If S1[NF] and S1[PF] have not been set since the last time the receive buffer was empty, the NOISY and PARITYE fields will be zero.

Address: Base address + Ch offset

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

UARTx_ED field descriptions

Field	Description
7 NOISY	The current received dataword contained in D and C3[R8] was received with noise. 0 The dataword was received without noise. 1 The data was received with noise.
6 PARITYE	The current received dataword contained in D and C3[R8] was received with a parity error. 0 The dataword was received without a parity error. 1 The dataword was received with a parity error.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

42.3.14 UART Modem Register (UARTx_MODEM)

The MODEM register controls options for setting the modem configuration.

Address: Base address + Dh offset

Bit	7	6	5	4	3	2	1	0
Read	0				RXRTSE	TXRTSPOL	TXRTSE	TXCTSE
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_MODEM field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 RXRTSE	Receiver request-to-send enable Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun. **NOTE:** Do not set both RXRTSE and TXRTSE. 0 The receiver has no effect on RTS. 1 RTS is deasserted if the number of characters in the receiver data register (FIFO) is equal to or greater than RWFIFO[RXWATER]. RTS is asserted when the number of characters in the receiver data register (FIFO) is less than RWFIFO[RXWATER]. See Hardware flow control
2 TXRTSPOL	Transmitter request-to-send polarity Controls the polarity of the transmitter RTS. TXRTSPOL does not affect the polarity of the receiver RTS. RTS will remain negated in the active low state unless TXRTSE is set. 0 Transmitter RTS is active low. 1 Transmitter RTS is active high.
1 TXRTSE	Transmitter request-to-send enable Controls RTS before and after a transmission.

Table continues on the next page...

UARTx_MODEM field descriptions (continued)

Field	Description
	0 The transmitter has no effect on RTS. 1 When a character is placed into an empty transmitter data buffer , RTS asserts one bit time before the start bit is transmitted. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. (FIFO) **NOTE:** Ensure that C2[TE] is asserted before assertion of this bit.
0 TXCTSE	Transmitter clear-to-send enable TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE. 0 CTS has no effect on the transmitter. 1 Enables clear-to-send operation. The transmitter checks the state of CTS each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the signal TXD remains in the mark state and transmission is delayed until CTS is asserted. Changes in CTS as a character is being sent do not affect its transmission.

42.3.15 UART FIFO Parameters (UARTx_PFIFO)

This register provides the ability for the programmer to turn on and off FIFO functionality. It also provides the size of the FIFO that has been implemented. This register may be read at any time. This register must be written only when C2[RE] and C2[TE] are cleared/not set and when the data buffer/FIFO is empty.

Address: Base address + 10h offset

Bit	7	6	5	4	3	2	1	0
Read	TXFE	TXFIFOSIZE			RXFE	RXFIFOSIZE		
Write								
Reset	0	*	*	*	0	*	*	*

* Notes:
- TXFIFOSIZE field: The reset value depends on whether the specific UART instance supports the FIFO and on the size of that FIFO. See the Chip Configuration details for more information on the FIFO size supported for each UART instance.
- RXFIFOSIZE field: The reset value depends on whether the specific UART instance supports the FIFO and on the size of that FIFO. See the Chip Configuration details for more information on the FIFO size supported for each UART instance.

UARTx_PFIFO field descriptions

Field	Description
7 TXFE	Transmit FIFO Enable When this field is set, the built in FIFO structure for the transmit buffer is enabled. The size of the FIFO structure is indicated by TXFIFOSIZE. If this field is not set, the transmit buffer operates as a FIFO of depth one dataword regardless of the value in TXFIFOSIZE. Both C2[TE] and C2[RE] must be cleared prior to changing this field. Additionally, TXFLUSH and RXFLUSH commands must be issued immediately after changing this field.

Table continues on the next page...

UARTx_PFIFO field descriptions (continued)

Field	Description
	0 Transmit FIFO is not enabled. Buffer is depth 1. (Legacy support). 1 Transmit FIFO is enabled. Buffer is depth indicated by TXFIFOSIZE.
6–4 TXFIFOSIZE	Transmit FIFO. Buffer Depth The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read only. 000 Transmit FIFO/Buffer depth = 1 dataword. 001 Transmit FIFO/Buffer depth = 4 datawords. 010 Transmit FIFO/Buffer depth = 8 datawords. 011 Transmit FIFO/Buffer depth = 16 datawords. 100 Transmit FIFO/Buffer depth = 32 datawords. 101 Transmit FIFO/Buffer depth = 64 datawords. 110 Transmit FIFO/Buffer depth = 128 datawords. 111 Reserved.
3 RXFE	Receive FIFO Enable When this field is set, the built in FIFO structure for the receive buffer is enabled. The size of the FIFO structure is indicated by the RXFIFOSIZE field. If this field is not set, the receive buffer operates as a FIFO of depth one dataword regardless of the value in RXFIFOSIZE. Both C2[TE] and C2[RE] must be cleared prior to changing this field. Additionally, TXFLUSH and RXFLUSH commands must be issued immediately after changing this field. 0 Receive FIFO is not enabled. Buffer is depth 1. (Legacy support) 1 Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE.
RXFIFOSIZE	Receive FIFO. Buffer Depth The maximum number of receive datawords that can be stored in the receive buffer before an overrun occurs. This field is read only. 000 Receive FIFO/Buffer depth = 1 dataword. 001 Receive FIFO/Buffer depth = 4 datawords. 010 Receive FIFO/Buffer depth = 8 datawords. 011 Receive FIFO/Buffer depth = 16 datawords. 100 Receive FIFO/Buffer depth = 32 datawords. 101 Receive FIFO/Buffer depth = 64 datawords. 110 Receive FIFO/Buffer depth = 128 datawords. 111 Reserved.

42.3.16 UART FIFO Control Register (UARTx_CFIFO)

This register provides the ability to program various control fields for FIFO operation. This register may be read or written at any time. Note that writing to TXFLUSH and RXFLUSH may result in data loss and requires careful action to prevent unintended/ unpredictable behavior. Therefore, it is recommended that TE and RE be cleared prior to flushing the corresponding FIFO.

Address: Base address + 11h offset

Bit	7	6	5	4	3	2	1	0
Read	0	0	0			RXOFE	TXOFE	RXUFE
Write	TXFLUSH	RXFLUSH						
Reset	0	0	0	0	0	0	0	0

UARTx_CFIFO field descriptions

Field	Description
7 TXFLUSH	Transmit FIFO/Buffer Flush Writing to this field causes all data that is stored in the transmit FIFO/buffer to be flushed. This does not affect data that is in the transmit shift register. 0 No flush operation occurs. 1 All data in the transmit FIFO/Buffer is cleared out.
6 RXFLUSH	Receive FIFO/Buffer Flush Writing to this field causes all data that is stored in the receive FIFO/buffer to be flushed. This does not affect data that is in the receive shift register. 0 No flush operation occurs. 1 All data in the receive FIFO/buffer is cleared out.
5–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 RXOFE	Receive FIFO Overflow Interrupt Enable When this field is set, the RXOF flag generates an interrupt to the host. 0 RXOF flag does not generate an interrupt to the host. 1 RXOF flag generates an interrupt to the host.
1 TXOFE	Transmit FIFO Overflow Interrupt Enable When this field is set, the TXOF flag generates an interrupt to the host. 0 TXOF flag does not generate an interrupt to the host. 1 TXOF flag generates an interrupt to the host.
0 RXUFE	Receive FIFO Underflow Interrupt Enable When this field is set, the RXUF flag generates an interrupt to the host.

Table continues on the next page...

UARTx_CFIFO field descriptions (continued)

Field	Description
	0 RXUF flag does not generate an interrupt to the host. 1 RXUF flag generates an interrupt to the host.

42.3.17 UART FIFO Status Register (UARTx_SFIFO)

This register provides status information regarding the transmit and receiver buffers/ FIFOs, including interrupt information. This register may be written to or read at any time.

Address: Base address + 12h offset

Bit	7	6	5	4	3	2	1	0
Read	TXEMPT	RXEMPT	0			RXOF	TXOF	RXUF
Write						w1c	w1c	w1c
Reset	1	1	0	0	0	0	0	0

UARTx_SFIFO field descriptions

Field	Description
7 TXEMPT	Transmit Buffer/FIFO Empty Asserts when there is no data in the Transmit FIFO/buffer. This field does not take into account data that is in the transmit shift register. 0 Transmit buffer is not empty. 1 Transmit buffer is empty.
6 RXEMPT	Receive Buffer/FIFO Empty Asserts when there is no data in the receive FIFO/Buffer. This field does not take into account data that is in the receive shift register. 0 Receive buffer is not empty. 1 Receive buffer is empty.
5–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 RXOF	Receiver Buffer Overflow Flag Indicates that more data has been written to the receive buffer than it can hold. This field will assert regardless of the value of CFIFO[RXOFE]. However, an interrupt will be issued to the host only if CFIFO[RXOFE] is set. This flag is cleared by writing a 1. 0 No receive buffer overflow has occurred since the last time the flag was cleared. 1 At least one receive buffer overflow has occurred since the last time the flag was cleared.
1 TXOF	Transmitter Buffer Overflow Flag

Table continues on the next page...

UARTx_SFIFO field descriptions (continued)

Field	Description
	Indicates that more data has been written to the transmit buffer than it can hold. This field will assert regardless of the value of CFIFO[TXOFE]. However, an interrupt will be issued to the host only if CFIFO[TXOFE] is set. This flag is cleared by writing a 1. 0 No transmit buffer overflow has occurred since the last time the flag was cleared. 1 At least one transmit buffer overflow has occurred since the last time the flag was cleared.
0 RXUF	Receiver Buffer Underflow Flag Indicates that more data has been read from the receive buffer than was present. This field will assert regardless of the value of CFIFO[RXUFE]. However, an interrupt will be issued to the host only if CFIFO[RXUFE] is set. This flag is cleared by writing a 1. 0 No receive buffer underflow has occurred since the last time the flag was cleared. 1 At least one receive buffer underflow has occurred since the last time the flag was cleared.

42.3.18 UART FIFO Transmit Watermark (UARTx_TWFIFO)

This register provides the ability to set a programmable threshold for notification of needing additional transmit data. This register may be read at any time but must be written only when C2[TE] is not set. Changing the value of the watermark will not clear the S1[TDRE] flag.

Address: Base address + 13h offset

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

UARTx_TWFIFO field descriptions

Field	Description
TXWATER	Transmit Watermark When the number of datawords in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt via S1[TDRE] or a DMA request via C5[TDMAS] is generated as determined by C5[TDMAS] and C2[TIE]. For proper operation, the value in TXWATER must be set to be less than the size of the transmit buffer/FIFO size as indicated by PFIFO[TXFIFOSIZE] and PFIFO[TXFE].

42.3.19 UART FIFO Transmit Count (UARTx_TCFIFO)

This is a read only register that indicates how many datawords are currently in the transmit buffer/FIFO. It may be read at any time.

Address: Base address + 14h offset

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

UARTx_TCFIFO field descriptions

Field	Description
TXCOUNT	Transmit Counter The value in this register indicates the number of datawords that are in the transmit FIFO/buffer. If a dataword is being transmitted, that is, in the transmit shift register, it is not included in the count. This value may be used in conjunction with PFIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer.

42.3.20 UART FIFO Receive Watermark (UARTx_RWFIFO)

This register provides the ability to set a programmable threshold for notification of the need to remove data from the receiver FIFO/buffer. This register may be read at any time but must be written only when C2[RE] is not asserted. Changing the value in this register will not clear S1[RDRF].

Address: Base address + 15h offset

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

UARTx_RWFIFO field descriptions

Field	Description
RXWATER	Receive Watermark When the number of datawords in the receive FIFO/buffer is equal to or greater than the value in this register field, an interrupt via S1[RDRF] or a DMA request via C5[RDMAS] is generated as determined by C5[RDMAS] and C2[RIE]. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by PFIFO[RXFIFOSIZE] and PFIFO[RXFE] and must be greater than 0.

42.3.21 UART FIFO Receive Count (UARTx_RCFIFO)

This is a read only register that indicates how many datawords are currently in the receive FIFO/buffer. It may be read at any time.

Address: Base address + 16h offset

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

UARTx_RCFIFO field descriptions

Field	Description
RXCOUNT	Receive Counter The value in this register indicates the number of datawords that are in the receive FIFO/buffer. If a dataword is being received, that is, in the receive shift register, it is not included in the count. This value may be used in conjunction with PFIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer.

42.4 Functional description

This section provides a complete functional description of the UART block.

The UART allows full duplex, asynchronous, NRZ serial communication between the CPU and remote devices, including other CPUs. The UART transmitter and receiver operate independently, although they use the same baud rate generator. The CPU monitors the status of the UART, writes the data to be transmitted, and processes received data.

42.4.1 Transmitter

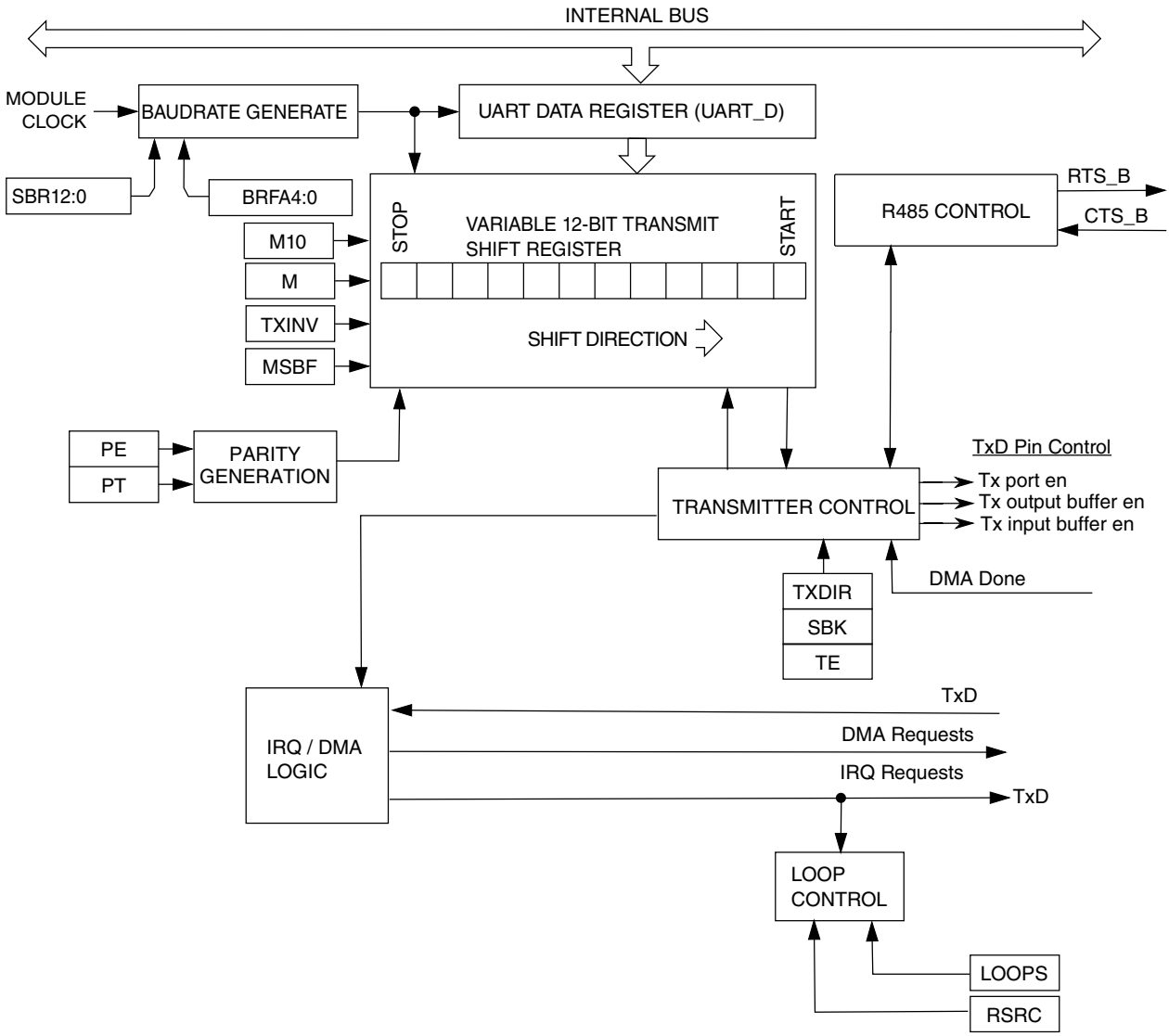


Figure 42-1. Transmitter Block Diagram

42.4.1.1 Transmitter character length

The UART transmitter can accommodate either 8, 9, or 10-bit data characters. The state of the C1[M] and C1[PE] bits and the C4[M10] bit determine the length of data characters. When transmitting 9-bit data, bit C3[T8] is the ninth bit (bit 8).

42.4.1.2 Transmission bit order

When S2[MSBF] is set, the UART automatically transmits the MSB of the data word as the first bit after the start bit. Similarly, the LSB of the data word is transmitted immediately preceding the parity bit, or the stop bit if parity is not enabled. All necessary bit ordering is handled automatically by the module. Therefore, the format of the data written to D for transmission is completely independent of the S2[MSBF] setting.

42.4.1.3 Character transmission

To transmit data, the MCU writes the data bits to the UART transmit buffer using UART data registers C3[T8] and D. Data in the transmit buffer is then transferred to the transmitter shift register as needed. The transmit shift register then shifts a frame out through the transmit data output signal after it has prefaced it with any required start and stop bits. The UART data registers, C3[T8] and D, provide access to the transmit buffer structure.

The UART also sets a flag, the transmit data register empty flag S1[TDRE], and generates an interrupt or DMA request (C5[TDMAS]) whenever the number of datawords in the transmit buffer is equal to or less than the value indicated by TWFIFO[TXWATER]. The transmit driver routine may respond to this flag by writing additional datawords to the transmit buffer using C3[T8]/D as space permits.

See Application information for specific programing sequences.

Setting C2[TE] automatically loads the transmit shift register with the following preamble:

Table 42-3. Transmit preamble length

BDH[SBNS]	C1[M]	C4[M10]	C1[PE]	Bits transmitted
0	0	—	—	10
1	0	—	—	11
0	1	0	—	11
1	1	0	—	12
0	1	1	1	12
1	1	1	1	13

After the preamble shifts out, control logic transfers the data from the D register into the transmit shift register. The transmitter automatically transmits the correct start bit and stop bit before and after the dataword. The number of stop bits transmitted after the dataword can be programmed using BDH[SBNS] field.

Hardware supports odd or even parity. When parity is enabled, the bit immediately preceding the stop bit is the parity bit.

When the transmit shift register is not transmitting a frame, the transmit data output signal goes to the idle condition, logic 1. If at any time software clears C2[TE], the transmitter enable signal goes low and the transmit signal goes idle.

If the software clears C2[TE] while a transmission is in progress, the character in the transmit shift register continues to shift out, provided S1[TC] was cleared during the data write sequence. To clear S1[TC], the S1 register must be read followed by a write to D register.

If S1[TC] is cleared during character transmission and C2[TE] is cleared, the transmission enable signal is deasserted at the completion of the current frame. Following this, the transmit data out signal enters the idle state even if there is data pending in the UART transmit data buffer. To ensure that all the data written in the FIFO is transmitted on the link before clearing C2[TE], wait for S1[TC] to set. Alternatively, the same can be achieved by setting TWFIFO[TXWATER] to 0x0 and waiting for S1[TDRE] to set.

42.4.1.4 Transmitting break characters

Setting C2[SBK] loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on C1[M], C1[PE], S2[BRK13], BDH[SBNS] and C4[M10]. See the following table.

Table 42-4. Transmit break character length

S2[BRK13]	BDH[SBNS]	C1[M]	C4[M10]	C1[PE]	Bits transmitted
0	0	0	—	—	10
0	1	0	—	—	11
0	0	1	0	—	11
0	1	1	0	—	12
0	0	1	1	1	12
0	1	1	1	1	13
1	0	0	—	—	13
1	0	1	—	—	14
1	1	0	—	—	15
1	1	1	—	—	16

As long as C2[SBK] is set, the transmitter logic continuously loads break characters into the transmit shift register. After the software clears C2[SBK], the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

NOTE

When queuing a break character, it will be transmitted following the completion of the data value currently being shifted out from the shift register. This means that, if data is queued in the data buffer to be transmitted, the break character preempts that queued data. The queued data is then transmitted after the break character is complete.

42.4.1.5 Idle characters

An idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on C1[M], C1[PE], BDH[SBNS] and C4[M10]. The preamble is a synchronizing idle character that begins the first transmission initiated after setting C2[TE].

If C2[TE] is cleared during a transmission, the transmit data output signal becomes idle after completion of the transmission in progress. Clearing and then setting C2[TE] during a transmission queues an idle character to be sent after the dataword currently being transmitted.

Note

When queuing an idle character, the idle character will be transmitted following the completion of the data value currently being shifted out from the shift register. This means that if data is queued in the data buffer to be transmitted, the idle character preempts that queued data. The queued data is then transmitted after the idle character is complete.

If C2[TE] is cleared and the transmission is completed, the UART is not the master of the TXD pin.

42.4.1.6 Hardware flow control

The transmitter supports hardware flow control by gating the transmission with the value of CTS. If the clear-to-send operation is enabled, the character is transmitted when CTS is asserted. If CTS is deasserted in the middle of a transmission with characters remaining in the receiver data buffer, the character in the shift register is sent and TXD remains in the mark state until CTS is reasserted.

If the clear-to-send operation is disabled, the transmitter ignores the state of CTS. Also, if the transmitter is forced to send a continuous low condition because it is sending a break character, the transmitter ignores the state of CTS regardless of whether the clear-to-send operation is enabled.

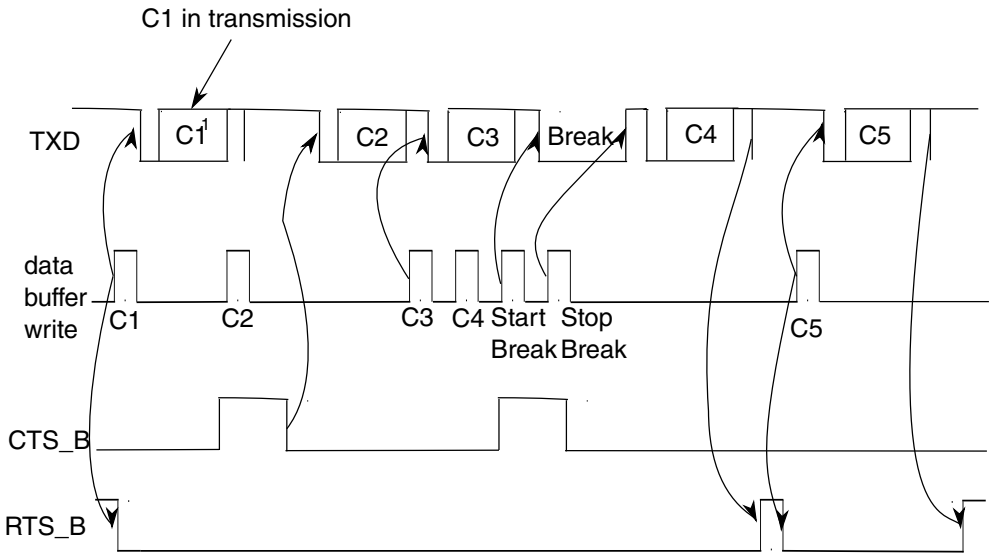
The transmitter's CTS signal can also be enabled even if the same UART receiver's RTS signal is disabled.

42.4.1.7 Transceiver driver enable

The transmitter can use RTS as an enable signal for the driver of an external transceiver. See Transceiver driver enable using RTS for details. If the request-to-send operation is enabled, when a character is placed into an empty transmitter data buffer, RTS asserts one bit time before the start bit is transmitted. RTS remains asserted for the whole time that the transmitter data buffer has any characters. RTS deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. Transmitting a break character also asserts RTS, with the same assertion and deassertion timing as having a character in the transmitter data buffer.

The transmitter's RTS signal asserts only when the transmitter is enabled. However, the transmitter's RTS signal is unaffected by its CTS signal. RTS will remain asserted until the transfer is completed, even if the transmitter is disabled mid-way through a data transfer.

The following figure shows the functional timing information for the transmitter. Along with the actual character itself, TXD shows the start bit. The stop bit is also indicated, with a dashed line if necessary.



1. Cn = transmit characters

Figure 42-2. Transmitter RTS and CTS timing diagram

42.4.2 Receiver

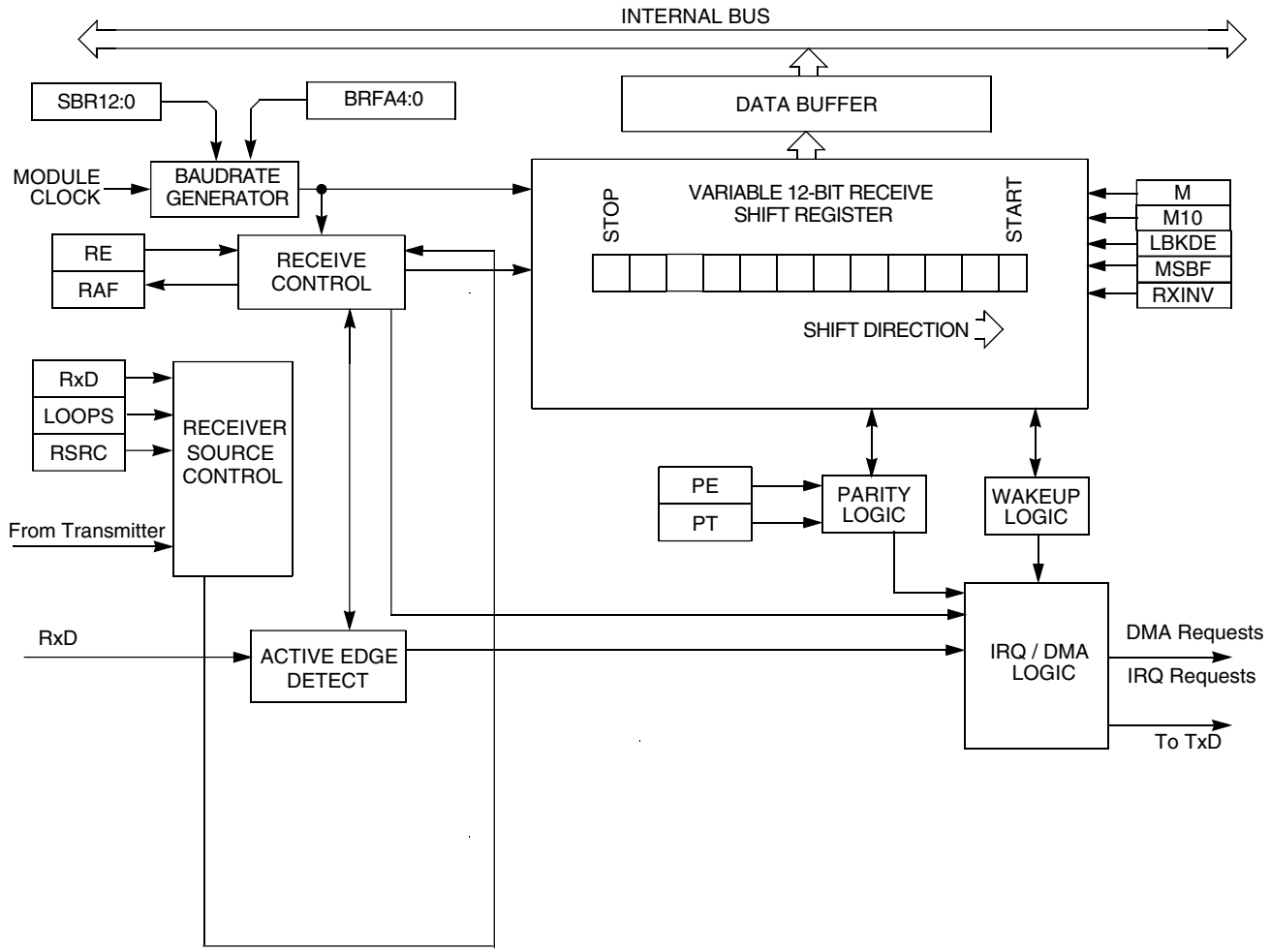


Figure 42-3. UART receiver block diagram

42.4.2.1 Receiver character length

The UART receiver can accommodate 8-, 9-, or 10-bit data characters. The states of C1[M], C1[PE], BDH[SBNS] and C4[M10] determine the length of data characters. When receiving 9 or 10-bit data, C3[R8] is the ninth bit (bit 8).

42.4.2.2 Receiver bit ordering

When S2[MSBF] is set, the receiver operates such that the first bit received after the start bit is the MSB of the dataword. Similarly, the bit received immediately preceding the parity bit, or the stop bit if parity is not enabled, is treated as the LSB for the dataword. All necessary bit ordering is handled automatically by the module. Therefore, the format of the data read from receive data buffer is completely independent of S2[MSBF].

42.4.2.3 Character reception

During UART reception, the receive shift register shifts a frame in from the unsynchronized receiver input signal. After a complete frame shifts into the receive shift register, the data portion of the frame transfers to the UART receive buffer. Additionally, the noise and parity error flags that are calculated during the receive process are also captured in the UART receive buffer. The receive data buffer is accessible via the D and C3[T8] registers. Additional received information flags regarding the receive dataword can be read in ED register. S1[RDRF] is set if the number of resulting datawords in the receive buffer is equal to or greater than the number indicated by RWFIFO[RXWATER]. If the C2[RIE] is also set, RDRF generates an RDRF interrupt request. Alternatively, by programming C5[RDMAS], a DMA request can be generated.

42.4.2.4 Data sampling

The receiver samples the unsynchronized receiver input signal 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 (see the following figure) is re-synchronized:

- After every start bit.
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0).

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

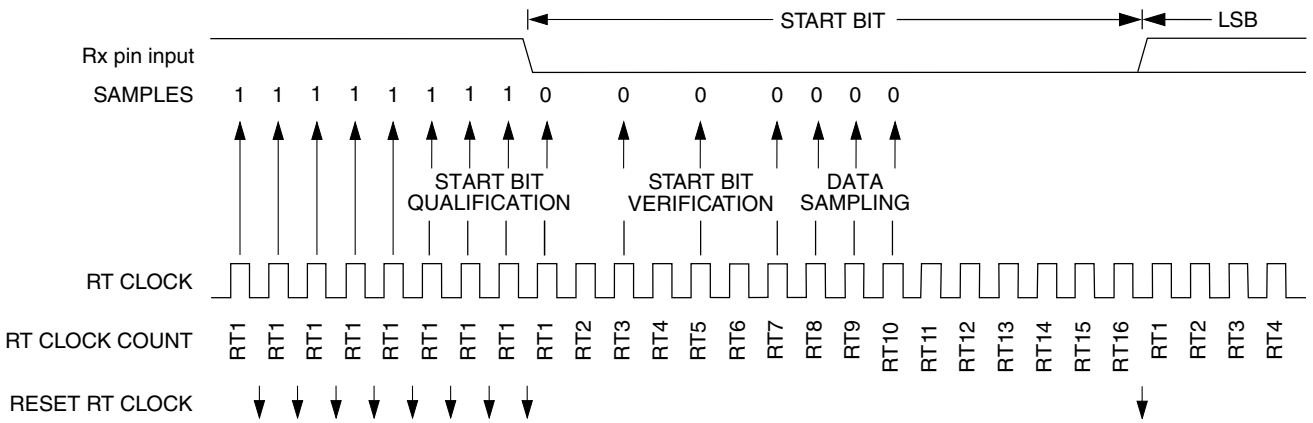


Figure 42-4. Receiver data sampling

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 42-5. 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 42-6. Data bit recovery

RT8, RT9, and RT10 samples	Data bit determination	Noise flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

Note

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (S1[NF]) is set and the receiver assumes that the bit is a start bit (logic 0).

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. The following table summarizes the results of the stop bit samples.

Table 42-7. 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.

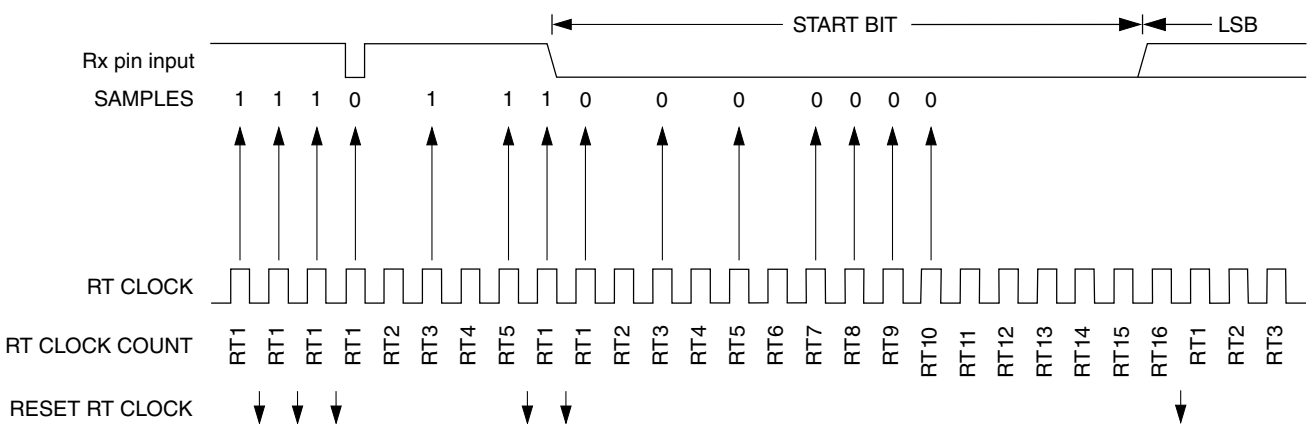


Figure 42-5. Start bit search example 1

In the following figure, 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.

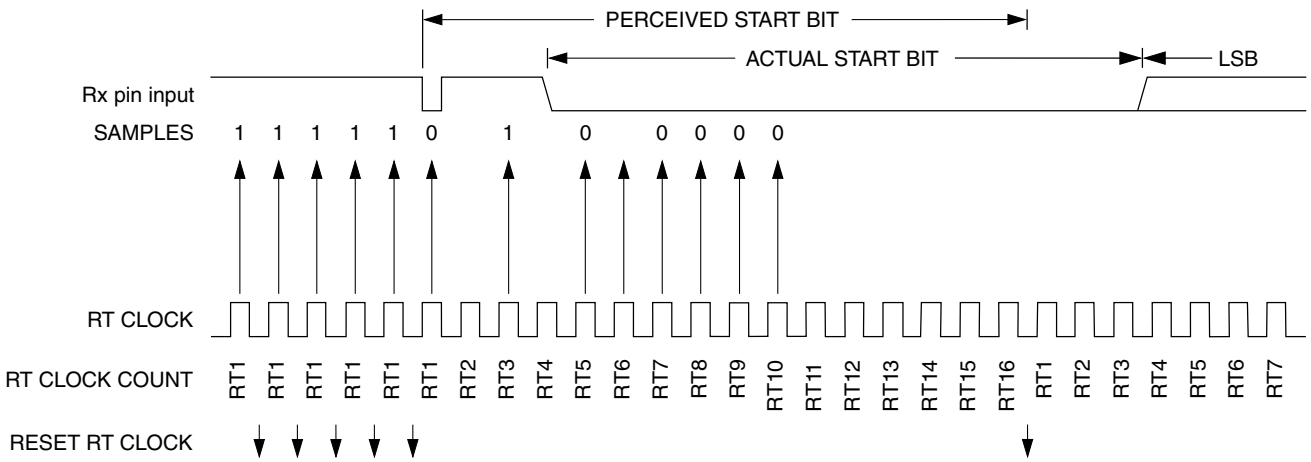


Figure 42-6. Start bit search example 2

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.

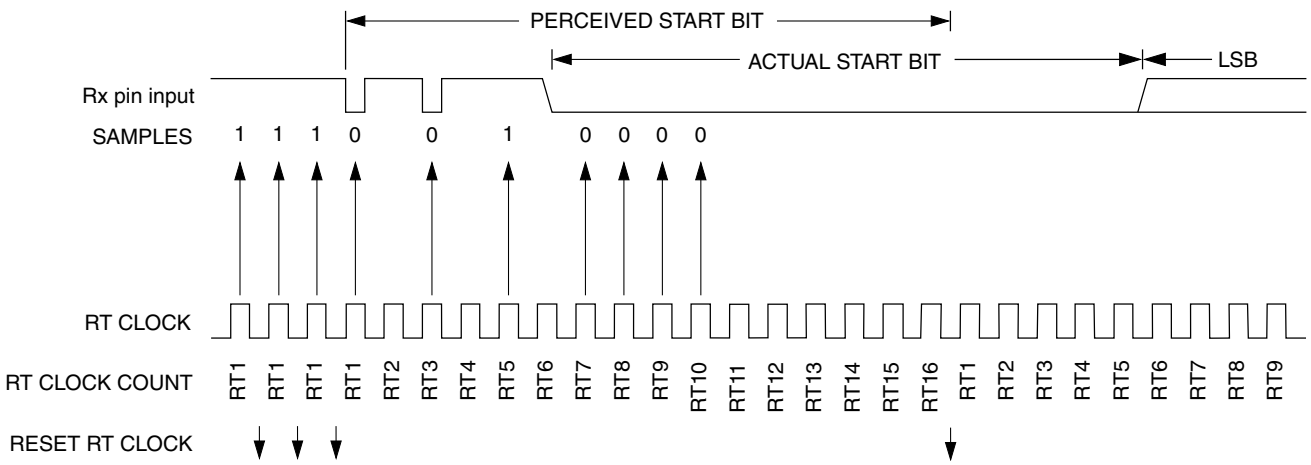


Figure 42-7. Start bit search example 3

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.

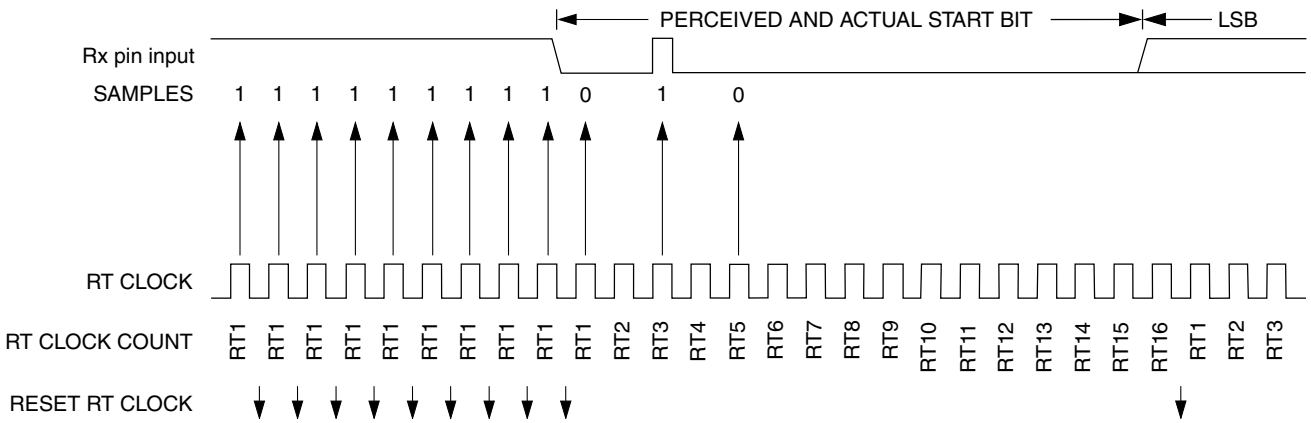


Figure 42-8. Start bit search example 4

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.

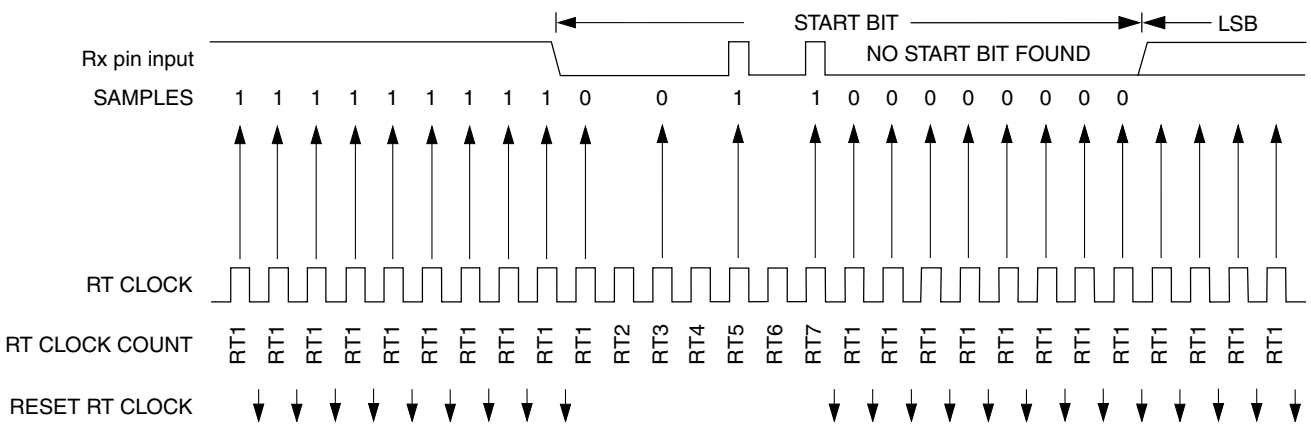


Figure 42-9. Start bit search example 5

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.

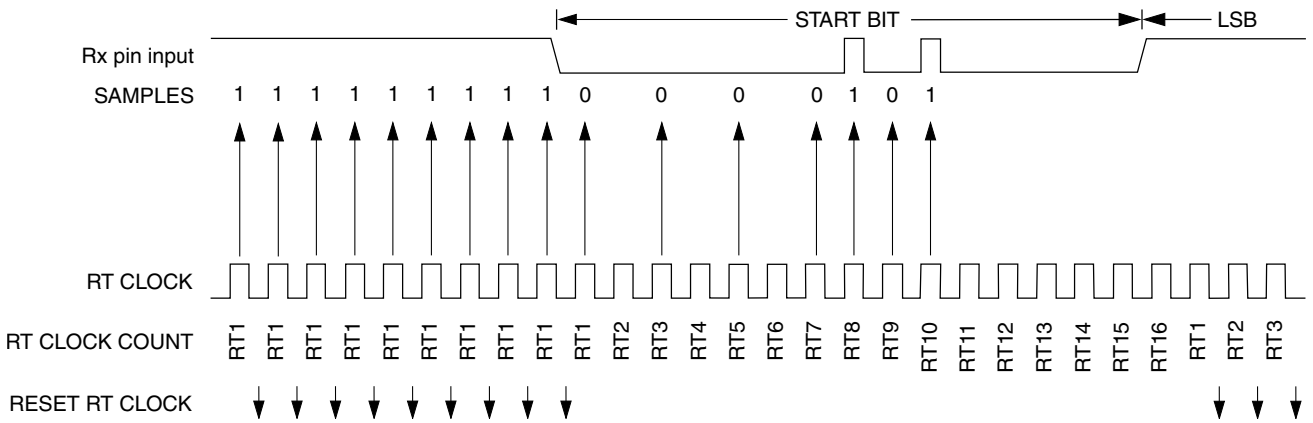


Figure 42-10. Start bit search example 6

42.4.2.5 Framing errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, S1[FE], if S2[LBKDE] is disabled. When S2[LBKDE] is disabled, a break character also sets the S1[FE] because a break character has no stop bit. S1[FE] is set at the same time that received data is placed in the receive data buffer.

42.4.2.6 Receiving break characters

The UART recognizes a break character when a start bit is followed by eight, nine, or ten logic 0 data bits and a logic 0 where the stop bit should be. Receiving a break character has these effects on UART registers:

- Sets the framing error flag, S1[FE].
- Writes an all 0 dataword to the data buffer, which may cause S1[RDRF] to set, depending on the watermark and number of values in the data buffer.
- May set the overrun flag, S1[OR], noise flag, S1[NF], parity error flag, S1[PE], or the receiver active flag, S2[RAF].

The detection threshold for a break character can be adjusted when using an internal oscillator in a LIN system by setting S2[LBKDE]. The UART break character detection threshold depends on C1[M], C1[PE], S2[LBKDE] and C4[M10]. See the following table.

Table 42-8. Receive break character detection threshold

LBKDE	SBNS	M	M10	PE	Threshold (bits)
0	0	0	—	—	10
0	1	0	—	—	11
0	0	1	0	—	11
0	1	1	0	—	12
0	0	1	1	1	12
0	1	1	1	1	13
1	—	0	—	—	11
1	—	1	—	—	12

While S2[LBKDE] is set, it will have these effects on the UART registers:

- Prevents S1[RDRF], S1[FE], S1[NF], and S1[PF] from being set. However, if they are already set, they will remain set.
- Sets the LIN break detect interrupt flag, S2[LBKDIF], if a LIN break character is received.

42.4.2.7 Hardware flow control

To support hardware flow control, the receiver can be programmed to automatically deassert and assert RTS.

- RTS remains asserted until the transfer is complete, even if the transmitter is disabled midway through a data transfer. See Transceiver driver enable using RTS for more details.
- If the receiver request-to-send functionality is enabled, the receiver automatically deasserts RTS if the number of characters in the receiver data register is equal to or greater than receiver data buffer's watermark, RWFIFO[RXWATER].
- The receiver asserts RTS when the number of characters in the receiver data register is less than the watermark. It is not affected if RDRF is asserted.
- Even if RTS is deasserted, the receiver continues to receive characters until the receiver data buffer is full or is overrun.
- If the receiver request-to-send functionality is disabled, the receiver RTS remains deasserted.

The following figure shows receiver hardware flow control functional timing. Along with the actual character itself, RXD shows the start bit. The stop bit can also indicated, with a dashed line, if necessary. The watermark is set to 2.

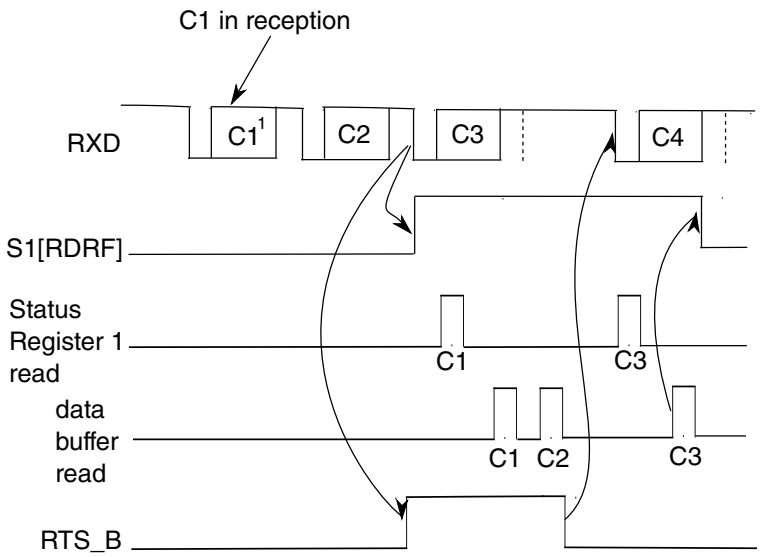


Figure 42-11. Receiver hardware flow control timing diagram

42.4.2.8 Baud rate tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stop bit samples are a logic 0.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames corrects a misalignment between transmitter bit times and receiver bit times.

42.4.2.8.1 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.

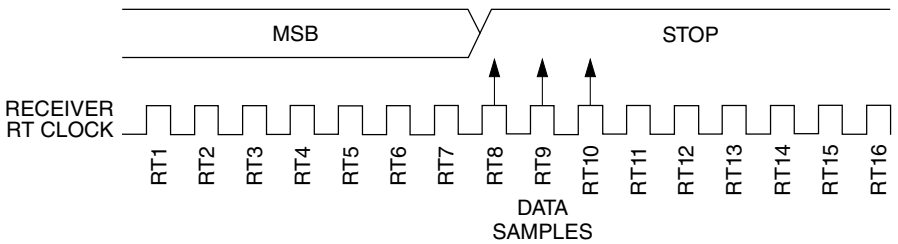


Figure 42-12. Slow data

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles (9 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 42-12, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 147 RT cycles (9 bit times × 16 RT cycles + 3 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:

$$((154 - 147) \div 154) \times 100 = 4.54\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 170 RT cycles (10 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 42-12, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 163 RT cycles (10 bit times × 16 RT cycles + 3 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$((170 - 163) \div 170) \times 100 = 4.12\%$$

42.4.2.8.2 Fast data tolerance

The following figure shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

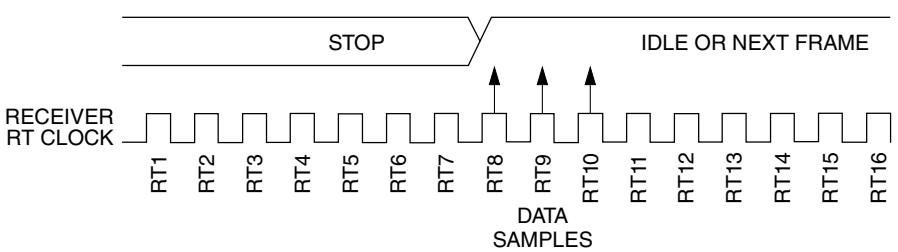


Figure 42-13. Fast data

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles (9 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 42-13, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 160 RT cycles (10 bit times × 16 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

$$((154 - 160) \div 154) \times 100 = 3.90\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 170 RT cycles (10 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the Figure 42-13, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 176 RT cycles (11 bit times × 16 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$((170 - 176) \div 170) \times 100 = 3.53\%$$

42.4.2.9 Receiver wakeup

C1[WAKE] determines how the UART is brought out of the standby state to process an incoming message. C1[WAKE] enables either idle line wakeup or address mark wakeup.

42.4.2.9.1 Idle input line wakeup (C1[WAKE] = 0)

In this wakeup method, an idle condition on the unsynchronized receiver input signal clears C2[RWU] and wakes the UART. The initial frame or frames of every message contain addressing information. All receivers evaluate the addressing information, and receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its C2[RWU] and return to the standby state. C2[RWU] remains set and the receiver remains in standby until another idle character appears on the unsynchronized receiver input signal.

Idle line wakeup requires that messages be separated by at least one idle character and that no message contains idle characters.

When C2[RWU] is 1 and S2[RWUID] is 0, the idle character that wakes the receiver does not set S1[IDLE] or the receive data register full flag, S1[RDRF]. The receiver wakes and waits for the first data character of the next message which is stored in the receive data buffer. When S2[RWUID] and C2[RWU] are set and C1[WAKE] is cleared, any idle condition sets S1[IDLE] and generates an interrupt if enabled.

42.4.2.9.2 Address mark wakeup (C1[WAKE] = 1)

In this wakeup method, a logic 1 in the bit position immediately preceding the stop bit of a frame clears C2[RWU] and wakes the UART. A logic 1 in the bit position immediately preceeding the stop bit 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 C2[RWU] and return to the standby state. C2[RWU] remains set and the receiver remains in standby until another address frame appears on the unsynchronized receiver input signal.

A logic 1 in the bit position immediately preceding the stop bit clears the receiver's C2[RWU] after the stop bit is received and places the received data into the receiver data buffer. Note that if Match Address operation is enabled i.e. C4[MAEN1] or C4[MAEN2] is set, then received frame is transferred to receive buffer only if the comparison matches.

Address mark wakeup allows messages to contain idle characters but requires that the bit position immediately preceding the stop bit be reserved for use in address frames.

If module is in standby mode and nothing triggers to wake the UART, no error flag is set even if an invalid error condition is detected on the receiving data line.

42.4.2.9.3 Match address operation

Match address operation is enabled when C4[MAEN1] or C4[MAEN2] is set. In this function, a frame received by the RX pin with a logic 1 in the bit position of the address mark is considered an address and is compared with the associated MA1 or MA2 register. The frame is transferred to the receive buffer, and S1[RDRF] is set, only if the comparison matches. All subsequent frames received with a logic 0 in the bit position of the address mark are considered to be data associated with the address and are transferred to the receive data buffer. If no marked address match occurs, then no transfer is made to the receive data buffer, and all following frames with logic 0 in the bit position of the address mark are also discarded. If both C4[MAEN1] and C4[MAEN2] are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Match address operation functions in the same way for both MA1 and MA2 registers. Note that the position of the address mark is the same as the Parity Bit when parity is enabled for 8 bit and 9 bit data formats.

- If only one of C4[MAEN1] and C4[MAEN2] is asserted, a marked address is compared only with the associated match register and data is transferred to the receive data buffer only on a match.
- If C4[MAEN1] and C4[MAEN2] are asserted, a marked address is compared with both match registers and data is transferred only on a match with either register.

42.4.3 Baud rate generation

A 13-bit modulus counter and a 5-bit fractional fine-adjust counter in the baud rate generator derive the baud rate for both the receiver and the transmitter. The value from 1 to 8191 written to SBR[12:0] determines the module clock divisor. The SBR bits are in the UART baud rate registers, BDH and BDL. The baud rate clock is synchronized with the module clock and drives the receiver. The fractional fine-adjust counter adds fractional delays to the baud rate clock to allow fine trimming of the baud rate to match the system baud rate. The transmitter is driven by the baud rate clock divided by 16. 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 module clock may not give the exact target frequency. This error can be reduced with the fine-adjust counter.
- Synchronization with the module clock can cause phase shift.

The Table 42-9 lists the available baud divisor fine adjust values.

UART baud rate = UART module clock / (16 × (SBR[12:0] + BRFD))

The following table lists some examples of achieving target baud rates with a module clock frequency of 10.2 MHz, with and without fractional fine adjustment.

Table 42-9. Baud rates (example: module clock = 10.2 MHz)

Bits SBR (decimal)	Bits BRFA	BRFD value	Receiver clock (Hz)	Transmitter clock (Hz)	Target Baud rate	Error (%)
17	00000	0	600,000.0	37,500.0	38,400	2.3
16	10011	19/32=0.59375	614,689.3	38,418.08	38,400	0.047

Table continues on the next page...

Table 42-9. Baud rates (example: module clock = 10.2 MHz) (continued)

Bits SBR (decimal)	Bits BRFA	BRFD value	Receiver clock (Hz)	Transmitter clock (Hz)	Target Baud rate	Error (%)
33	00000	0	309,090.9	19,318.2	19,200	0.62
33	00110	6/32=0.1875	307,344.6	19,209.04	19,200	0.047
66	00000	0	154,545.5	9659.1	9600	0.62
133	00000	0	76,691.7	4793.2	4800	0.14
266	00000	0	38,345.9	2396.6	2400	0.14
531	00000	0	19,209.0	1200.6	1200	0.11
1062	00000	0	9604.5	600.3	600	0.05
2125	00000	0	4800.0	300.0	300	0.00
4250	00000	0	2400.0	150.0	150	0.00
5795	00000	0	1760.1	110.0	110	0.00

Table 42-10. Baud rate fine adjust

BRFA	Baud Rate Fractional Divisor (BRFD)
0 0 0 0 0	0/32 = 0
0 0 0 0 1	1/32 = 0.03125
0 0 0 1 0	2/32 = 0.0625
0 0 0 1 1	3/32 = 0.09375
0 0 1 0 0	4/32 = 0.125
0 0 1 0 1	5/32 = 0.15625
0 0 1 1 0	6/32 = 0.1875
0 0 1 1 1	7/32 = 0.21875
0 1 0 0 0	8/32 = 0.25
0 1 0 0 1	9/32 = 0.28125
0 1 0 1 0	10/32 = 0.3125
0 1 0 1 1	11/32 = 0.34375
0 1 1 0 0	12/32 = 0.375
0 1 1 0 1	13/32 = 0.40625
0 1 1 1 0	14/32 = 0.4375
0 1 1 1 1	15/32 = 0.46875
1 0 0 0 0	16/32 = 0.5
1 0 0 0 1	17/32 = 0.53125
1 0 0 1 0	18/32 = 0.5625
1 0 0 1 1	19/32 = 0.59375
1 0 1 0 0	20/32 = 0.625
1 0 1 0 1	21/32 = 0.65625
1 0 1 1 0	22/32 = 0.6875

Table continues on the next page...

Table 42-10. Baud rate fine adjust (continued)

BRFA	Baud Rate Fractional Divisor (BRFD)
1 0 1 1 1	23/32 = 0.71875
1 1 0 0 0	24/32 = 0.75
1 1 0 0 1	25/32 = 0.78125
1 1 0 1 0	26/32 = 0.8125
1 1 0 1 1	27/32 = 0.84375
1 1 1 0 0	28/32 = 0.875
1 1 1 0 1	29/32 = 0.90625
1 1 1 1 0	30/32 = 0.9375
1 1 1 1 1	31/32 = 0.96875

42.4.4 Data format

Each data character is contained in a frame that includes a start bit and a stop bit. The rest of the data format depends upon C1[M], C1[PE], S2[MSBF], BDH[SBNS] and C4[M10].

42.4.4.1 Eight-bit configuration

Clearing C1[M] configures the UART for 8-bit data characters, that is, eight bits are memory mapped in D. A frame with eight data bits has a total of 10 bits (This becomes 11 bits if BDH[SBNS] = 1). The most significant bit of the eight data bits can be used as an address mark to wake the receiver. If the most significant bit is used in this way, then it serves as an address or data indication, leaving the remaining seven bits as actual data. When C1[PE] is set, the eighth data bit is automatically calculated as the parity bit. See the following table.

Table 42-11. Configuration of 8-bit data format

UART_C1[PE]	Start bit	Data bits	Address bits	Parity bits	Stop bit
0	1	8	0	0	1
0	1	7	1[1]	0	1
1	1	7	0	1	1

1. The address bit identifies the frame as an address character. See Receiver wakeup.

NOTE

In the last column of the above table, the number of stop bits become 2 when BDH[SBNS] is set.

42.4.4.2 Nine-bit configuration

When C1[M] is set and C4[M10] is cleared and BDH[SBNS] is cleared, the UART is configured for 9-bit data characters. If C1[PE] is enabled, the ninth bit is either C3[T8/R8] or the internally generated parity bit. This results in a frame consisting of a total of 11 bits. In the event that the ninth data bit is selected to be C3[T8], it will remain unchanged after transmission and can be used repeatedly without rewriting it, unless the value needs to be changed. This feature may be useful when the ninth data bit is being used as an address mark.

When C1[M] and C4[M10] are set and BDH[SBNS] is cleared, the UART is configured for 9-bit data characters, but the frame consists of a total of 12 bits. The 12 bits include the start and stop bits, the 9 data character bits, and a tenth internal data bit. Note that if C4[M10] is set, C1[PE] must also be set. In this case, the tenth bit is the internally generated parity bit. The ninth bit can either be used as an address mark or a ninth data bit.

See the following table.

Table 42-12. Configuration of 9-bit data formats

C1[PE]	UC1[M]	C1[M10]	Start bit	Data bits	Address bits	Parity bits	Stop bit
0	0	0	See Eight-bit configuration				
0	0	1	Invalid configuration				
0	1	0	1	9	0	0	1
0	1	0	1	8	1 [1]	0	1
0	1	1	Invalid Configuration				
1	0	0	See Eight-bit configuration				
1	0	1	Invalid Configuration				
1	1	0	1	8	0	1	1
1	1	1	1	9	0	1	1
1	1	1	1	8	1[1]	1	1

1. The address bit identifies the frame as an address character.

NOTE

In the last column of the above table, the number of stop bits become 2 when BDH[SBNS] is set.

Note

Unless in 9-bit mode with M10 set, do not use address mark wakeup with parity enabled.

42.4.4.3 Timing examples

Timing examples of these configurations in the NRZ mark/space data format are illustrated in the following figures. The timing examples show all of the configurations in the following sub-sections along with the LSB and MSB first variations. This section explains the data formats available assuming single stop bit mode is selected.

42.4.4.3.1 Eight-bit format with parity disabled

The most significant bit can be used for address mark wakeup.

Figure 42-14. Eight bits of data with LSB first

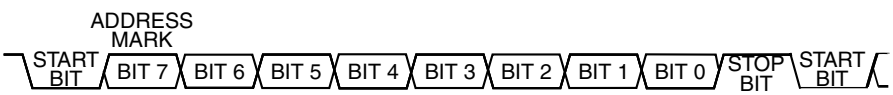


Figure 42-15. Eight bits of data with MSB first

42.4.4.3.2 Eight-bit format with parity enabled

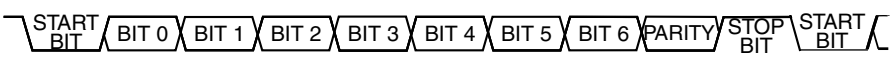


Figure 42-16. Seven bits of data with LSB first and parity

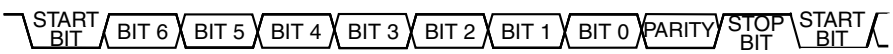


Figure 42-17. Seven bits of data with MSB first and parity

42.4.4.3.3 Nine-bit format with parity disabled

The most significant bit can be used for address mark wakeup.

Figure 42-18. Nine bits of data with LSB first

Figure 42-19. Nine bits of data with MSB first

42.4.4.3.4 Nine-bit format with parity enabled

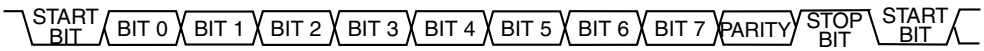


Figure 42-20. Eight bits of data with LSB first and parity

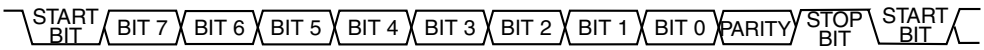


Figure 42-21. Eight bits of data with MSB first and parity

42.4.4.3.5 Non-memory mapped tenth bit for parity

The most significant memory-mapped bit can be used for address mark wakeup.

Figure 42-22. Nine bits of data with LSB first and parity

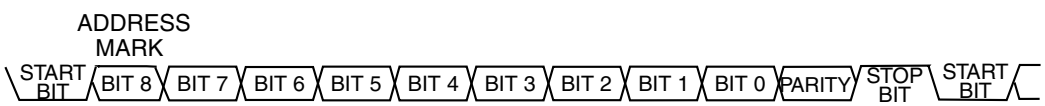


Figure 42-23. Nine bits of data with MSB first and parity

42.4.5 Single-wire operation

Normally, the UART uses two pins for transmitting and receiving. In single wire operation, the RXD pin is disconnected from the UART and the UART implements a half-duplex serial connection. The UART uses the TXD pin for both receiving and transmitting.

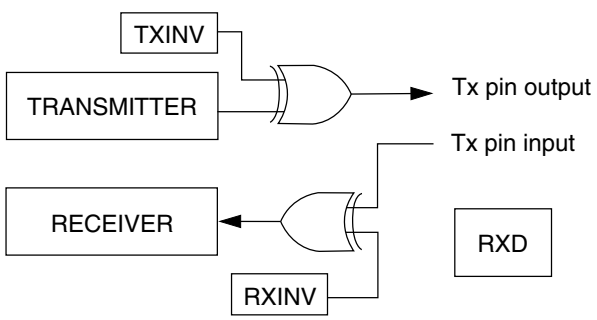


Figure 42-24. Single-wire operation (C1[LOOPS] = 1, C1[RSRC] = 1)

Enable single wire operation by setting C1[LOOPS] and the receiver source field, C1[RSRC]. Setting C1[LOOPS] disables the path from the unsynchronized receiver input signal to the receiver. Setting C1[RSRC] connects the receiver input to the output of the TXD pin driver. Both the transmitter and receiver must be enabled (C2[TE] = 1 and C2[RE] = 1).

42.4.6 Loop operation

In loop operation, the transmitter output goes to the receiver input. The unsynchronized receiver input signal is disconnected from the UART.

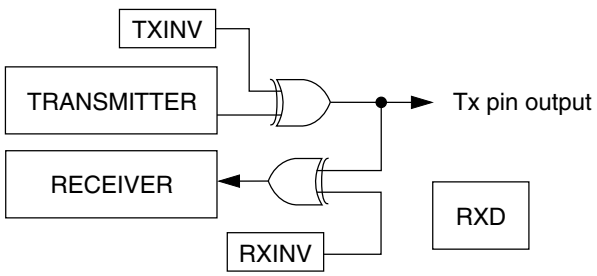


Figure 42-25. Loop operation (C1[LOOPS] = 1, C1[RSRC] = 0)

Enable loop operation by setting C1[LOOPS] and clearing C1[RSRC]. Setting C1[LOOPS] disables the path from the unsynchronized receiver input signal to the receiver. Clearing C1[RSRC] connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (C2[TE] = 1 and C2[RE] = 1).

42.5 Reset

All registers reset to a particular value are indicated in Memory map and registers.

42.6 System level interrupt sources

There are several interrupt signals that are sent from the UART. The following table lists the interrupt sources generated by the UART. The local enables for the UART interrupt sources are described in this table. Details regarding the individual operation of each interrupt are contained under various sub-sections of Memory map and registers. However, RXEDGIF description also outlines additional details regarding the RXEDGIF interrupt because of its complexity of operation. Any of the UART interrupt requests listed in the table can be used to bring the CPU out of Wait mode.

Table 42-13. UART interrupt sources

Interrupt Source	Flag	Local enable	DMA select
Transmitter	TDRE	TIE	TDMAS = 0
Transmitter	TC	TCIE	-
Receiver	IDLE	ILIE	-
Receiver	RDRF	RIE	RDMAS = 0
Receiver	LBKDIF	LBKDIE	-
Receiver	RXEDGIF	RXEDGIE	-
Receiver	OR	ORIE	-
Receiver	NF	NEIE	-
Receiver	FE	FEIE	-
Receiver	PF	PEIE	-
Receiver	RXUF	RXUFE	-
Transmitter	TXOF	TXOFE	-

42.6.1 RXEDGIF description

S2[RXEDGIF] is set when an active edge is detected on the RxD pin. Therefore, the active edge can be detected only when in two wire mode. A RXEDGIF interrupt is generated only when S2[RXEDGIF] is set. If RXEDGIE is not enabled before S2[RXEDGIF] is set, an interrupt is not generated.

42.6.1.1 RxD edge detect sensitivity

Edge sensitivity can be software programmed to be either falling or rising. The polarity of the edge sensitivity is selected using S2[RXINV]. To detect the falling edge, S2[RXINV] is programmed to 0. To detect the rising edge, S2[RXINV] is programmed to 1.

Synchronizing logic is used prior to detect edges. Prior to detecting an edge, the receive data on RxD input must be at the deasserted logic level. A falling edge is detected when the RxD input signal is seen as a logic 1 (the deasserted level) during one module clock cycle, and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input is seen as a logic 0 during one module clock cycle and then a logic 1 during the next cycle.

42.6.1.2 Clearing RXEDGIF interrupt request

Writing a logic 1 to S2[RXEDGIF] immediately clears the RXEDGIF interrupt request even if the RxD input remains asserted. S2[RXEDGIF] remains set if another active edge is detected on RxD while attempting to clear S2[RXEDGIF] by writing a 1 to it.

42.6.1.3 Exit from low-power modes

The receive input active edge detect circuit is still active on low power modes (Wait and Stop). An active edge on the receive input brings the CPU out of low power mode if the interrupt is not masked (S2[RXEDGIF] = 1).

42.7 DMA operation

In the transmitter, S1[TDRE] can be configured to assert a DMA transfer request. In the receiver, S1[RDRF], can be configured to assert a DMA transfer request. The following table shows the configuration field settings required to configure each flag for DMA operation.

Table 42-14. DMA configuration

Flag	Request enable bit	DMA select bit
TDRE	TIE = 1	TDMAS = 1
RDRF	RIE = 1	RDMAS = 1

When a flag is configured for a DMA request, its associated DMA request is asserted when the flag is set. When S1[RDRF] is configured as a DMA request, the clearing mechanism of reading S1, followed by reading D, does not clear the associated flag. The DMA request remains asserted until an indication is received that the DMA transactions are done. When this indication is received, the flag bit and the associated DMA request is cleared. If the DMA operation failed to remove the situation that caused the DMA request, another request is issued.

42.8 Application information

This section describes the UART application information.

42.8.1 Transmit/receive data buffer operation

The UART has independent receive and transmit buffers. The size of these buffers may vary depending on the implementation of the module. The implemented size of the buffers is a fixed constant via PFIFO[TXFIFOSIZE] and PFIFO[RXFIFOSIZE]. Additionally, legacy support is provided that allows for the FIFO structure to operate as a depth of one. This is the default/reset behavior of the module and can be adjusted using the PFIFO[RXFE] and PFIFO[TXFE] bits. Individual watermark levels are also provided for transmit and receive.

There are multiple ways to ensure that a data block, which is a set of characters, has completed transmission. These methods include:

1. Set TXFIFO[TXWATER] to 0. TDRE asserts when there is no further data in the transmit buffer. Alternatively the S1[TC] flag can be used to indicate when the transmit shift register is also empty.
2. Poll TCFIFO[TXCOUNT]. Assuming that only data for a data block has been put into the data buffer, when TCFIFO[TXCOUNT] = 0, all data has been transmitted or is in the process of transmission.
3. S1[TC] can be monitored. When S1[TC] asserts, it indicates that all data has been transmitted and there is no data currently being transmitted in the shift register.

42.8.2 Initialization sequence

To initiate a UART transmission:

1. Configure the UART.
    a. Select a baud rate. Write this value to the UART baud registers (BDH/L) to begin the baud rate generator. Remember that the baud rate generator is disabled when the baud rate is zero. Writing to the BDH has no effect without also writing to BDL.
    b. Write to C1 to configure word length, parity, and other configuration bits (LOOPS, RSRC, M, WAKE, ILT, PE, and PT). Write to C4, MA1, and MA2 to configure.
    c. Enable the transmitter, interrupts, receiver, and wakeup as required, by writing to C2 (TIE, TCIE, RIE, ILIE, TE, RE, RWU, and SBK), S2 (MSBF and BRK13), and C3 (ORIE, NEIE, PEIE, and FEIE). A preamble or idle character is then shifted out of the transmitter shift register.

2. Transmit procedure for each byte.
   a. Monitor S1[TDRE] by reading S1 or responding to the TDRE interrupt. The amount of free space in the transmit buffer directly using TCFIFO[TXCOUNT] can also be monitored.
   b. If the TDRE flag is set, or there is space in the transmit buffer, write the data to be transmitted to (C3[T8]/D). A new transmission will not result until data exists in the transmit buffer.
3. Repeat step 2 for each subsequent transmission.

Note

During normal operation, S1[TDRE] is set when the shift register is loaded with the next data to be transmitted from the transmit buffer and the number of datawords contained in the transmit buffer is less than or equal to the value in TWFIFO[TXWATER]. This occurs 9/16ths of a bit time after the start of the stop bit of the previous frame.

To separate messages with preambles with minimum idle line time, use this sequence between messages.

1. Write the last dataword of the first message to C3[T8]/D.
2. Wait for S1[TDRE] to go high with TWFIFO[TXWATER] = 0, indicating the transfer of the last frame to the transmit shift register.
3. Queue a preamble by clearing and then setting C2[TE].
4. Write the first and subsequent datawords of the second message to C3[T8]/D.

42.8.3 Overrun (OR) flag implications

To be flexible, the overrun flag (OR) operates slight differently depending on the mode of operation. There may be implications that need to be carefully considered. This section clarifies the behavior and the resulting implications. Regardless of mode, if a dataword is received while S1[OR] is set, S1[RDRF] and S1[IDLE] are blocked from asserting. If S1[RDRF] or S1[IDLE] were previously asserted, they will remain asserted until cleared.

42.8.3.1 Overrun operation

The assertion of S1[OR] indicates that a significant event has occurred. The assertion indicates that received data has been lost because there was a lack of room to store it in the data buffer. Therefore, while S1[OR] is set, no further data is stored in the data buffer until S1[OR] is cleared. This ensures that the application will be able to handle the overrun condition.

In most applications, because the total amount of lost data is known, the application will attempt to return the system to a known state. Before S1[OR] is cleared, all received data will be dropped. For this, the software does the following.

1. Remove data from the receive data buffer. This could be done by reading data from the data buffer and processing it if the data in the FIFO was still valuable when the overrun event occurred, or using CFIFO[RXFLUSH] to clear the buffer.
2. Clear S1[OR]. Note that if data was cleared using CFIFO[RXFLUSH], then clearing S1[OR] will result in SFIFO[RXUF] asserting. This is because the only way to clear S1[OR] requires reading additional information from the FIFO. Care should be taken to disable the SFIFO[RXUF] interrupt prior to clearing the OR flag and then clearing SFIFO[RXUF] after the OR flag has been cleared.

When LIN break detect (LBKDE) is asserted, S1[OR] has significantly different behavior than in other modes. S1[OR] will be set, regardless of how much space is actually available in the data buffer, if a LIN break character has been detected and the corresponding flag, S2[LBKDIF], is not cleared before the first data character is received after S2[LBKDIF] asserted. This behavior is intended to allow the software sufficient time to read the LIN break character from the data buffer to ensure that a break character was actually detected. The checking of the break character was used on some older implementations and is therefore supported for legacy reasons. Applications that do not require this checking can simply clear S2[LBKDIF] without checking the stored value to ensure it is a break character.

42.8.4 Match address registers

The two match address registers allow a second match address function for a broadcast or general call address to the serial bus, as an example.

42.8.5 Modem feature

This section describes the modem features.

42.8.5.1 Ready-to-receive using RTS

To help to stop overrun of the receiver data buffer, the RTS signal can be used by the receiver to indicate to another UART that it is ready to receive data. The other UART can send the data when its CTS signal is asserted. This handshaking conforms to the TIA-232-E standard. A transceiver is necessary if the required voltage levels of the communication link do not match the voltage levels of the UART's RTS and CTS signals.

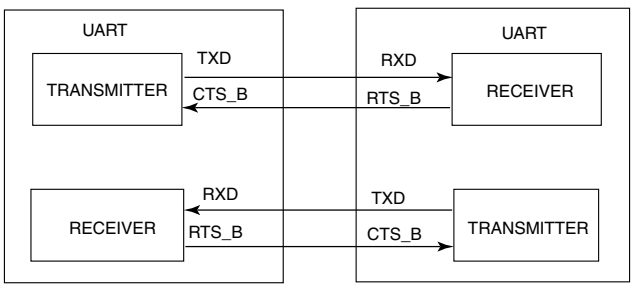


Figure 42-26. Ready-to-receive

The transmitter's CTS signal can be used for hardware flow control whether its RTS signal is used for hardware flow control, transceiver driver enable, or not at all.

42.8.5.2 Transceiver driver enable using RTS

RS-485 is a multiple drop communication protocol in which the UART transceiver's driver is 3-stated unless the UART is driving. The RTS signal can be used by the transmitter to enable the driver of a transceiver. The polarity of RTS can be matched to the polarity of the transceiver's driver enable signal. See the following figure.

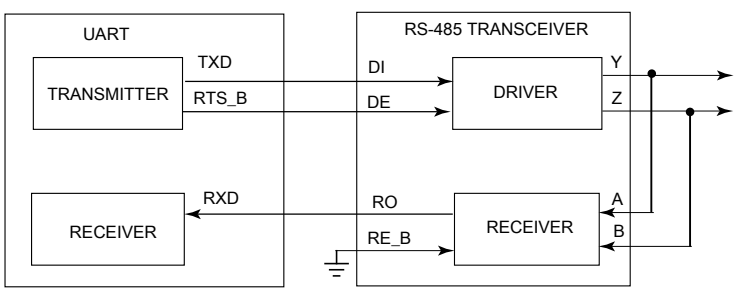


Figure 42-27. Transceiver driver enable using RTS

In the figure, the receiver enable signal is asserted. Another option for this connection is to connect RTS_B to both DE and RE_B. The transceiver's receiver is disabled while driving. A pullup can pull RXD to a non-floating value during this time. This option can be refined further by operating the UART in single wire mode, freeing the RXD pin for other uses.

42.8.6 Legacy and reverse compatibility considerations

Recent versions of the UART have added several new features. Whenever reasonably possible, reverse compatibility was maintained. However, in some cases this was either not feasible or the behavior was deemed as not intended. This section describes several differences to legacy operation that resulted from these recent enhancements. If application code from previous versions is used, it must be reviewed and modified to take the following items into account. Depending on the application code, additional items that are not listed here may also need to be considered.

1. Various reserved registers and register bits are used, such as, MSFB and M10.
2. This module now generates an error when invalid address spaces are used.
3. While documentation indicated otherwise, in some cases it was possible for S1[IDLE] to assert even if S1[OR] was set.
4. S1[OR] will be set only if the data buffer (FIFO) does not have sufficient room. Previously, the data buffer was always a fixed size of one and the S1[OR] flag would set so long as S1[RDRF] was set even if there was room in the data buffer. While the clearing mechanism has remained the same for S1[RDRF], keeping the OR flag assertion tied to the RDRF event rather than the data buffer being full would have greatly reduced the usefulness of the buffer when its size is larger than one.
5. Previously, when C2[RWU] was set (and WAKE = 0), the IDLE flag could reassert up to every bit period causing an interrupt and requiring the host processor to reassert C2[RWU]. This behavior has been modified. Now, when C2[RWU] is set (and WAKE = 0), at least one non-idle bit must be detected before an idle can be detected.

Chapter 43
General-Purpose Input/Output (GPIO)

43.1 Introduction

NOTE

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

The general-purpose input and output (GPIO) module is accessible via the peripheral bus and also communicates to the processor core via a zero wait state interface (IOPORT) for maximum pin performance. The GPIO registers support 8-bit, 16-bit or 32-bit accesses.

The GPIO data direction and output data registers control the direction and output data of each pin when the pin is configured for the GPIO function. The GPIO input data register displays the logic value on each pin when the pin is configured for any digital function, provided the corresponding Port Control and Interrupt module for that pin is enabled.

Efficient bit manipulation of the general-purpose outputs is supported through the addition of set, clear, and toggle write-only registers for each port output data register.

43.1.1 Features

Features of the GPIO module include:

- Port Data Input register visible in all digital pin-multiplexing modes
- Port Data Output register with corresponding set/clear/toggle registers
- Port Data Direction register
- Zero wait state access to GPIO registers through IOPORT

NOTE

The GPIO module is clocked by system clock.

43.1.2 Modes of operation

The following table depicts different modes of operation and the behavior of the GPIO module in these modes.

Table 43-1. Modes of operation

Modes of operation	Description
Run	The GPIO module operates normally.
Wait	The GPIO module operates normally.
Stop	The GPIO module is disabled.
Debug	The GPIO module operates normally.

43.1.3 GPIO signal descriptions

Table 43-2. GPIO signal descriptions

GPIO signal descriptions	Description	I/O
PORTA31–PORTA0	General-purpose input/output	I/O
PORTB31–PORTB0	General-purpose input/output	I/O
PORTC31–PORTC0	General-purpose input/output	I/O
PORTD31–PORTD0	General-purpose input/output	I/O
PORTE31–PORTE0	General-purpose input/output	I/O

NOTE

Not all pins within each port are implemented on each device. See the chapter on signal multiplexing for the number of GPIO ports available in the device.

43.1.3.1 Detailed signal description

Table 43-3. GPIO interface-detailed signal descriptions

Signal	I/O	Description	
PORTA31–PORTA0	I/O	General-purpose input/output	
PORTB31–PORTB0 PORTC31–PORTC0		State meaning	Asserted: The pin is logic 1. Deasserted: The pin is logic 0.
PORTD31–PORTD0 PORTE31–PORTE0		Timing	Assertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time

Table 43-3. GPIO interface-detailed signal descriptions

Signal	I/O	Description	
			and input may be asserted asynchronously to the system clock. Deassertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time and input may be asserted asynchronously to the system clock.

NOTE

Not all pins within each port are implemented on each device. See the chapter on signal multiplexing for the number of GPIO ports available in the device.

43.2 Memory map and register definition

Any read or write access to the GPIO memory space that is outside the valid memory map results in a bus error.

NOTE

For simplicity, each GPIO port's registers appear with the same width of 32 bits, corresponding to 32 pins. The actual number of pins per port (and therefore the number of usable control bits per port register) is chip-specific. Refer to the Chip Configuration chapter to see the exact control bits for the non-identical port instance.

GPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_F000	Port Data Output Register (GPIOA_PDOR)	32	R/W	0000_0000h	43.2.1/1151
400F_F004	Port Set Output Register (GPIOA_PSOR)	32	W (always reads 0)	0000_0000h	43.2.2/1152
400F_F008	Port Clear Output Register (GPIOA_PCOR)	32	W (always reads 0)	0000_0000h	43.2.3/1152

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_F00C	Port Toggle Output Register (GPIOA_PTOR)	32	W (always reads 0)	0000_0000h	43.2.4/1153
400F_F010	Port Data Input Register (GPIOA_PDIR)	32	R	0000_0000h	43.2.5/1153
400F_F014	Port Data Direction Register (GPIOA_PDDR)	32	R/W	0000_0000h	43.2.6/1154
400F_F040	Port Data Output Register (GPIOB_PDOR)	32	R/W	0000_0000h	43.2.1/1151
400F_F044	Port Set Output Register (GPIOB_PSOR)	32	W (always reads 0)	0000_0000h	43.2.2/1152
400F_F048	Port Clear Output Register (GPIOB_PCOR)	32	W (always reads 0)	0000_0000h	43.2.3/1152
400F_F04C	Port Toggle Output Register (GPIOB_PTOR)	32	W (always reads 0)	0000_0000h	43.2.4/1153
400F_F050	Port Data Input Register (GPIOB_PDIR)	32	R	0000_0000h	43.2.5/1153
400F_F054	Port Data Direction Register (GPIOB_PDDR)	32	R/W	0000_0000h	43.2.6/1154
400F_F080	Port Data Output Register (GPIOC_PDOR)	32	R/W	0000_0000h	43.2.1/1151
400F_F084	Port Set Output Register (GPIOC_PSOR)	32	W (always reads 0)	0000_0000h	43.2.2/1152
400F_F088	Port Clear Output Register (GPIOC_PCOR)	32	W (always reads 0)	0000_0000h	43.2.3/1152
400F_F08C	Port Toggle Output Register (GPIOC_PTOR)	32	W (always reads 0)	0000_0000h	43.2.4/1153
400F_F090	Port Data Input Register (GPIOC_PDIR)	32	R	0000_0000h	43.2.5/1153
400F_F094	Port Data Direction Register (GPIOC_PDDR)	32	R/W	0000_0000h	43.2.6/1154
400F_F0C0	Port Data Output Register (GPIOD_PDOR)	32	R/W	0000_0000h	43.2.1/1151
400F_F0C4	Port Set Output Register (GPIOD_PSOR)	32	W (always reads 0)	0000_0000h	43.2.2/1152
400F_F0C8	Port Clear Output Register (GPIOD_PCOR)	32	W (always reads 0)	0000_0000h	43.2.3/1152
400F_F0CC	Port Toggle Output Register (GPIOD_PTOR)	32	W (always reads 0)	0000_0000h	43.2.4/1153
400F_F0D0	Port Data Input Register (GPIOD_PDIR)	32	R	0000_0000h	43.2.5/1153
400F_F0D4	Port Data Direction Register (GPIOD_PDDR)	32	R/W	0000_0000h	43.2.6/1154
400F_F100	Port Data Output Register (GPIOE_PDOR)	32	R/W	0000_0000h	43.2.1/1151

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_F104	Port Set Output Register (GPIOE_PSOR)	32	W (always reads 0)	0000_0000h	43.2.2/1152
400F_F108	Port Clear Output Register (GPIOE_PCOR)	32	W (always reads 0)	0000_0000h	43.2.3/1152
400F_F10C	Port Toggle Output Register (GPIOE_PTOR)	32	W (always reads 0)	0000_0000h	43.2.4/1153
400F_F110	Port Data Input Register (GPIOE_PDIR)	32	R	0000_0000h	43.2.5/1153
400F_F114	Port Data Direction Register (GPIOE_PDDR)	32	R/W	0000_0000h	43.2.6/1154

43.2.1 Port Data Output Register (GPIOx_PDOR)

This register configures the logic levels that are driven on each general-purpose output pins.

NOTE

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

Address: Base address + 0h 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 W	PDO																															
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

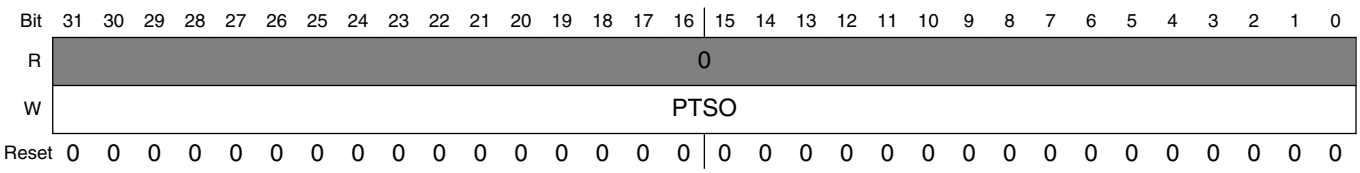
GPIOx_PDOR field descriptions

Field	Description
PDO	Port Data Output Register bits for unbonded pins return a undefined value when read. 0 Logic level 0 is driven on pin, provided pin is configured for general-purpose output. 1 Logic level 1 is driven on pin, provided pin is configured for general-purpose output.

43.2.2 Port Set Output Register (GPIOx_PSOR)

This register configures whether to set the fields of the PDOR.

Address: Base address + 4h offset



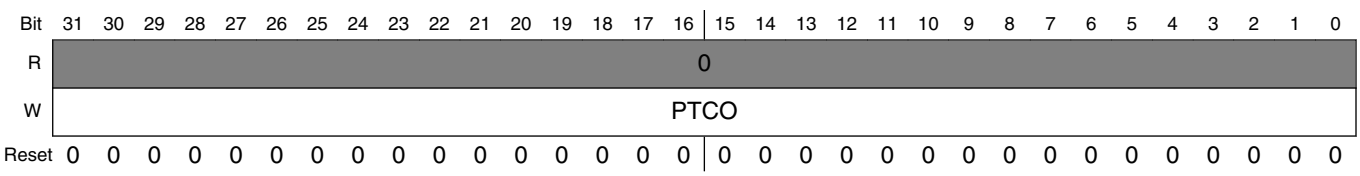
GPIOx_PSOR field descriptions

Field	Description
PTSO	Port Set Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to logic 1.

43.2.3 Port Clear Output Register (GPIOx_PCOR)

This register configures whether to clear the fields of PDOR.

Address: Base address + 8h offset

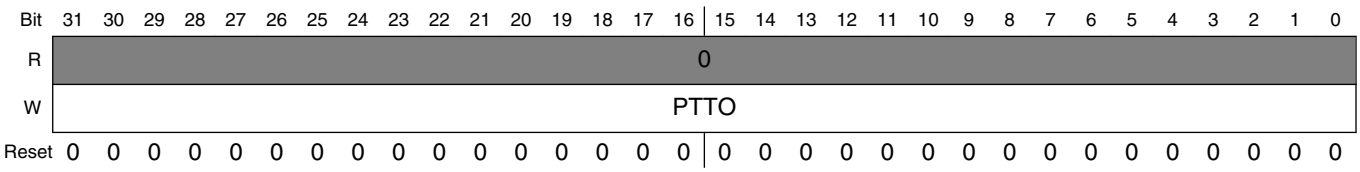


GPIOx_PCOR field descriptions

Field	Description
PTCO	Port Clear Output Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is cleared to logic 0.

43.2.4 Port Toggle Output Register (GPIOx_PTOR)

Address: Base address + Ch offset



GPIOx_PTOR field descriptions

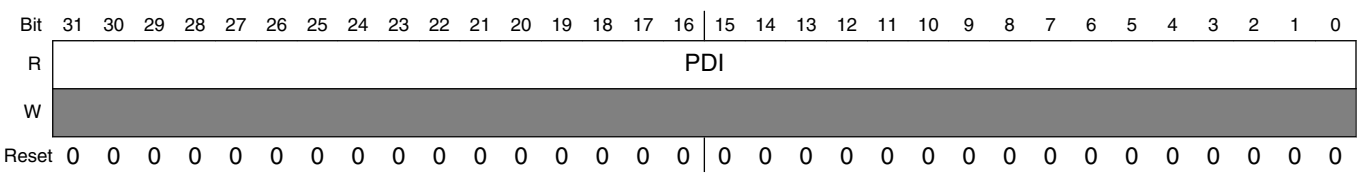
Field	Description
PTTO	Port Toggle Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to the inverse of its existing logic state.

43.2.5 Port Data Input Register (GPIOx_PDIR)

NOTE

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

Address: Base address + 10h offset



GPIOx_PDIR field descriptions

Field	Description
PDI	Port Data Input Reads 0 at the unimplemented pins for a particular device. Pins that are not configured for a digital function read 0. If the Port Control and Interrupt module is disabled, then the corresponding bit in PDIR does not update. 0 Pin logic level is logic 0, or is not configured for use by digital function. 1 Pin logic level is logic 1.

43.2.6 Port Data Direction Register (GPIOx_PDDR)

The PDDR configures the individual port pins for input or output.

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 W	PDD																															
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

GPIOx_PDDR field descriptions

Field	Description
PDD	Port Data Direction Configures individual port pins for input or output. 0 Pin is configured as general-purpose input, for the GPIO function. 1 Pin is configured as general-purpose output, for the GPIO function.

43.3 FGPIO memory map and register definition

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800_0000.

Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. This aliased Fast GPIO memory map is called FGPIO.

Any read or write access to the FGPIO memory space that is outside the valid memory map results in a bus error. All register accesses complete with zero wait states, except error accesses which complete with one wait state.

NOTE

For simplicity, each FGPIO port's registers appear with the same width of 32 bits, corresponding to 32 pins. The actual number of pins per port (and therefore the number of usable control bits per port register) is chip-specific. Refer to the Chip Configuration chapter to see the exact control bits for the non-identical port instance.

FGPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F800_0000	Port Data Output Register (FGPIOA_PDOR)	32	R/W	0000_0000h	43.3.1/1156
F800_0004	Port Set Output Register (FGPIOA_PSOR)	32	W (always reads 0)	0000_0000h	43.3.2/1157
F800_0008	Port Clear Output Register (FGPIOA_PCOR)	32	W (always reads 0)	0000_0000h	43.3.3/1157
F800_000C	Port Toggle Output Register (FGPIOA_PTOR)	32	W (always reads 0)	0000_0000h	43.3.4/1158
F800_0010	Port Data Input Register (FGPIOA_PDIR)	32	R	0000_0000h	43.3.5/1158
F800_0014	Port Data Direction Register (FGPIOA_PDDR)	32	R/W	0000_0000h	43.3.6/1159
F800_0040	Port Data Output Register (FGPIOB_PDOR)	32	R/W	0000_0000h	43.3.1/1156
F800_0044	Port Set Output Register (FGPIOB_PSOR)	32	W (always reads 0)	0000_0000h	43.3.2/1157
F800_0048	Port Clear Output Register (FGPIOB_PCOR)	32	W (always reads 0)	0000_0000h	43.3.3/1157
F800_004C	Port Toggle Output Register (FGPIOB_PTOR)	32	W (always reads 0)	0000_0000h	43.3.4/1158
F800_0050	Port Data Input Register (FGPIOB_PDIR)	32	R	0000_0000h	43.3.5/1158
F800_0054	Port Data Direction Register (FGPIOB_PDDR)	32	R/W	0000_0000h	43.3.6/1159
F800_0080	Port Data Output Register (FGPIOC_PDOR)	32	R/W	0000_0000h	43.3.1/1156
F800_0084	Port Set Output Register (FGPIOC_PSOR)	32	W (always reads 0)	0000_0000h	43.3.2/1157
F800_0088	Port Clear Output Register (FGPIOC_PCOR)	32	W (always reads 0)	0000_0000h	43.3.3/1157
F800_008C	Port Toggle Output Register (FGPIOC_PTOR)	32	W (always reads 0)	0000_0000h	43.3.4/1158
F800_0090	Port Data Input Register (FGPIOC_PDIR)	32	R	0000_0000h	43.3.5/1158
F800_0094	Port Data Direction Register (FGPIOC_PDDR)	32	R/W	0000_0000h	43.3.6/1159
F800_00C0	Port Data Output Register (FGPIOD_PDOR)	32	R/W	0000_0000h	43.3.1/1156
F800_00C4	Port Set Output Register (FGPIOD_PSOR)	32	W (always reads 0)	0000_0000h	43.3.2/1157
F800_00C8	Port Clear Output Register (FGPIOD_PCOR)	32	W (always reads 0)	0000_0000h	43.3.3/1157

Table continues on the next page...

FGPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F800_00CC	Port Toggle Output Register (FGPIOD_PTOR)	32	W (always reads 0)	0000_0000h	43.3.4/1158
F800_00D0	Port Data Input Register (FGPIOD_PDIR)	32	R	0000_0000h	43.3.5/1158
F800_00D4	Port Data Direction Register (FGPIOD_PDDR)	32	R/W	0000_0000h	43.3.6/1159
F800_0100	Port Data Output Register (FGPIOE_PDOR)	32	R/W	0000_0000h	43.3.1/1156
F800_0104	Port Set Output Register (FGPIOE_PSOR)	32	W (always reads 0)	0000_0000h	43.3.2/1157
F800_0108	Port Clear Output Register (FGPIOE_PCOR)	32	W (always reads 0)	0000_0000h	43.3.3/1157
F800_010C	Port Toggle Output Register (FGPIOE_PTOR)	32	W (always reads 0)	0000_0000h	43.3.4/1158
F800_0110	Port Data Input Register (FGPIOE_PDIR)	32	R	0000_0000h	43.3.5/1158
F800_0114	Port Data Direction Register (FGPIOE_PDDR)	32	R/W	0000_0000h	43.3.6/1159

43.3.1 Port Data Output Register (FGPIOx_PDOR)

This register configures the logic levels that are driven on each general-purpose output pins.

Address: Base address + 0h 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 W	PDO																															
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

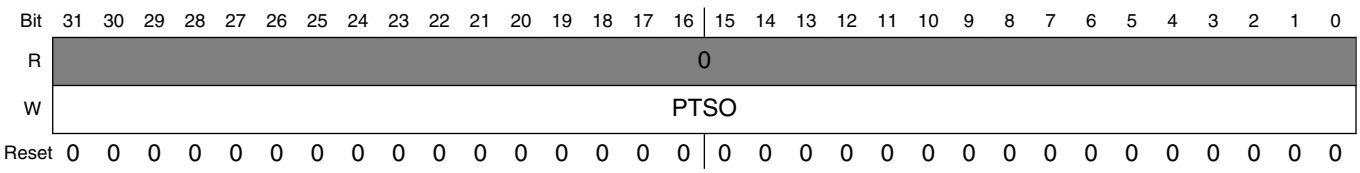
FGPIOx_PDOR field descriptions

Field	Description
PDO	Port Data Output Unimplemented pins for a particular device read as zero. 0 Logic level 0 is driven on pin, provided pin is configured for general-purpose output. 1 Logic level 1 is driven on pin, provided pin is configured for general-purpose output.

43.3.2 Port Set Output Register (FGPIOx_PSOR)

This register configures whether to set the fields of the PDOR.

Address: Base address + 4h offset



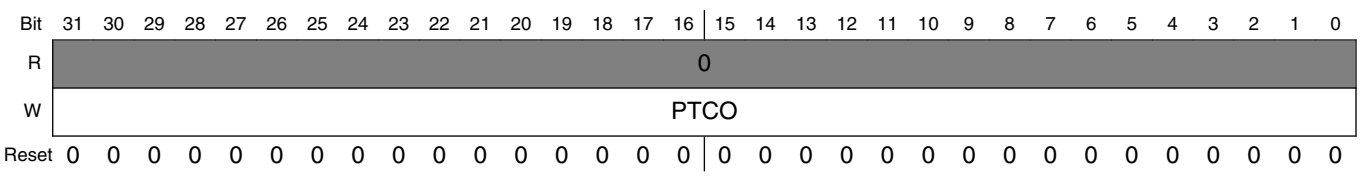
FGPIOx_PSOR field descriptions

Field	Description
PTSO	Port Set Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to logic 1.

43.3.3 Port Clear Output Register (FGPIOx_PCOR)

This register configures whether to clear the fields of PDOR.

Address: Base address + 8h offset

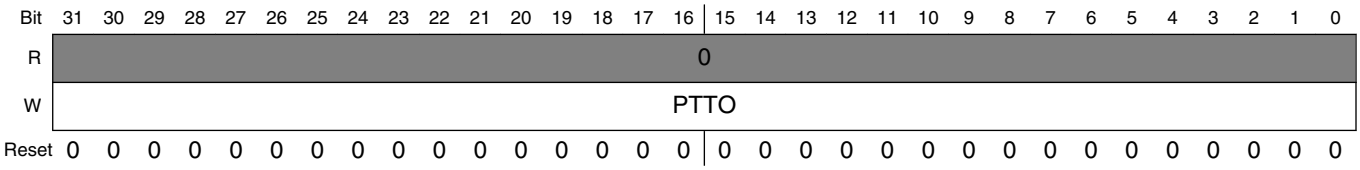


FGPIOx_PCOR field descriptions

Field	Description
PTCO	Port Clear Output Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is cleared to logic 0.

43.3.4 Port Toggle Output Register (FGPIOx_PTOR)

Address: Base address + Ch offset

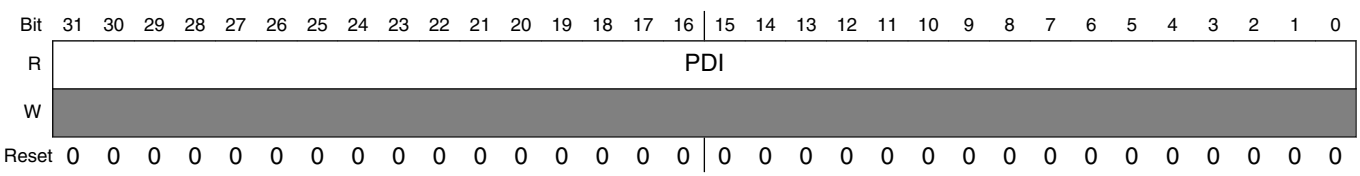


FGPIOx_PTOR field descriptions

Field	Description
PTTO	Port Toggle Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to the inverse of its existing logic state.

43.3.5 Port Data Input Register (FGPIOx_PDIR)

Address: Base address + 10h offset



FGPIOx_PDIR field descriptions

Field	Description
PDI	Port Data Input Reads 0 at the unimplemented pins for a particular device. Pins that are not configured for a digital function read 0. If the Port Control and Interrupt module is disabled, then the corresponding bit in PDIR does not update. 0 Pin logic level is logic 0, or is not configured for use by digital function. 1 Pin logic level is logic 1.

43.3.6 Port Data Direction Register (FGPIOx_PDDR)

The PDDR configures the individual port pins for input or output.

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 W	PDD																															
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

FGPIOx_PDDR field descriptions

Field	Description
PDD	Port Data Direction Configures individual port pins for input or output. 0 Pin is configured as general-purpose input, for the GPIO function. 1 Pin is configured as general-purpose output, for the GPIO function.

43.4 Functional description

43.4.1 General-purpose input

The logic state of each pin is available via the Port Data Input registers, provided the pin is configured for a digital function and the corresponding Port Control and Interrupt module is enabled.

The Port Data Input registers return the synchronized pin state after any enabled digital filter in the Port Control and Interrupt module. The input pin synchronizers are shared with the Port Control and Interrupt module, so that if the corresponding Port Control and Interrupt module is disabled, then synchronizers are also disabled. This reduces power consumption when a port is not required for general-purpose input functionality.

43.4.2 General-purpose output

The logic state of each pin can be controlled via the port data output registers and port data direction registers, provided the pin is configured for the GPIO function. The following table depicts the conditions for a pin to be configured as input/output.

If	Then

Table continues on the next page...

A pin is configured for the GPIO function and the corresponding port data direction register bit is clear.	The pin is configured as an input.
A pin is configured for the GPIO function and the corresponding port data direction register bit is set.	The pin is configured as an output and and the logic state of the pin is equal to the corresponding port data output register.

To facilitate efficient bit manipulation on the general-purpose outputs, pin data set, pin data clear, and pin data toggle registers exist to allow one or more outputs within one port to be set, cleared, or toggled from a single register write.

The corresponding Port Control and Interrupt module does not need to be enabled to update the state of the port data direction registers and port data output registers including the set/clear/toggle registers.

43.4.3 IOPORT

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800_0000. Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. If the DMA attempts to access the GPIO registers on the same cycle as an IOPORT access, then the DMA access will stall until any IOPORT accesses have completed.

Chapter 44
Release Notes for Revision 4

44.1 General changes throughout document

- Removed the SPIx_SREX register.

44.2 About This Document chapter changes

- No substantial content changes

44.3 Introduction chapter changes

- Added KMS part numbers to the table "Orderable part numbers summary."
- Added the Kinetis Motor Suite module information to the section "Module functional categories."

44.4 Chip Configuration chapter changes

- Added the following sections to support Kinetis Motor Suite (KMS):
  - Kinetis Motor Suite configuration
  - Library protection
  - Flash protection
  - Flash Configuration and Access Control Values
  - Configuration of the production MCUs
- Updated the following sections:
  - KV11x Flash Memory Sizes
  - Flash security
  - FTFA_FOPT Register
  - KMS configuration

44.5 Memory Map chapter changes

- No substantial content changes

44.6 Clock Distribution chapter changes

- No substantial content changes

44.7 Reset and Boot chapter changes

- No substantial content changes

44.8 Power Management chapter changes

- No substantial content changes

44.9 Security chapter changes

- No substantial content changes

44.10 Debug chapter changes

- No substantial content changes

44.11 Signal Multiplexing and Signal Descriptions chapter changes

- Added the table "Flex CAN signal descriptions

44.12 Port Control and Interrupts (PORT) changes

- For the PCR registers, changed the access type of the reserved bit 5 to R/W.
- In the "Pin Control Register n (PORT_PCRn)" section, clarified the last paragraph in the note.

44.13 SIM changes

- No substantial content changes

44.14 Kinetis Flash Bootloader changes

- In "Start-up Process" section, replaced "The flashloader initializes the .data and .bss sections" with "The flashloader's temporary working area in RAM is initialized".
- In "GetProperty command" section, added mentions of internal memory regions, and 2 footnotes.
- In FlashEraseAll command section, in "FlashEraseAll Command Packet Format" table, added MemoryID parameter to command packet parameters
- In "FillMemory command" section, corrected missing text for "Writing to flash requires the start address to be"
- In "Get/SetProperty Command Properties" section, table "Properties used by Get/SetProperty Commands, sorted by Value", updated the description of RAMStartAddress and RAMSizeInBytes properties.

44.15 System Mode Controller changes (SMC)

- No substantial content changes

44.16 PMC changes

- No substantial content changes

44.17 LLWU changes

- No substantial content changes

44.18 Reset Control Module changes (RCM)

- No substantial content changes

44.19 BME configuration changes

- No substantial content changes

44.20 MMDVSQ changes

- No substantial content changes

44.21 MCM changes

- No substantial content changes

44.22 MTB configuration changes

- No substantial content changes

44.23 Crossbar switch module changes

- Features :
  - Replaced "64-bit data bus" with "Up to single-clock 32-bit transfer".
- Removed bullet beginning with, "Operation at a 1-to-1 clock frequency..." from Features.
- General operation : Removed paragraph beginning with "A master is given control of the targeted slave..." and the following list, beginning with "A higher priority master has...".
- Initialization/application information :
  - Changed wording of sentence about arbitration scheme.

44.24 AIPS-Lite module changes

- No substantial content changes

44.25 DMAMUX module changes

- Updated the offset address of registers in the code example given in the section "Enabling and configuring sources"
- Removed the address information of CHCFG1 register (base address + 0x01) and CHCFG8 (base address + 0x08) from the section "Enabling and configuring sources."

44.26 eDMA module changes

- Edited Introduction.
- Dynamic channel linking : Added cross-reference to TCD structure.
- Features : Replaced "optional" with other wording to improve clarity.
- Changed note in CLM field description of DMA_CR register.
- Error Status Register (DMA_ES) : Added two causes of channel errors to list in register description.
- DMA_TCD*n*_CSR[BWC]: Removed note from field description.
- DMA_TCD*n*_CSR[ACTIVE]: Changed access from RW to RO.
- Error Status Register (DMA_ES) : Removed bullet about uncorrectable TCD SRAM errors.
- eDMA initialization : In bullet beginning with "Enable any hardware service requests..." replaced "ERQH and ERQL registers" with "ERQ register."
- Error Status Register (DMA_ES) : In description, replaced "See the Error Reporting and Handling section" with "See Fault reporting and handling.

44.27 EWM changes

- Substantial changes throughout the chapter which include:
  - Changed the term "service" to "refresh" throughout the chapter to better explain the EWM refresh mechanism.
  - Updated the block diagram.
  - Updated the table "EWM refresh mechanism".
  - Editorial changes

44.28 WDOG changes

- No substantial content changes

44.29 MCG changes

- No substantial content changes

44.30 OSC changes

- No substantial content changes

44.31 FMC changes

- No substantial content changes

44.32 FTFA changes

- Add ACCERR check for sector size larger than segment size in Error Handling table for RD1XA and ERSXA commands
- Modify FSEC[MEEN] register field description
- Modify Flash Commands by Mode table entries for Read 1s All Blocks and Erase All Blocks commands
- Add Read 1s All Execute-only Segments and Erase All Execute-only Segments commands; modify list of Margin Read Commands
- Add reference to AN5112 in Flash Access Protection
- Add ACCERR check for mode/security in Error Handling table for Verify Backdoor Access Key and Read 1s All Blocks commands
- Change column heading from Byte to Offset Address in configuration field description table
- Add suggestion to bit poll FSTAT[CCIF] for command completion in Generic flash command write sequence flowchart
- Clarify that ACCERR and FPVIOL flags must be clear before ERSSUSP can be set in Suspending an Erase Flash Sector Operation
- Remove erroneous reference to the flash configuration field in Suspending an Erase Flash Sector Operation
- Specify minimum time of 4.3 msec between request to resume and suspend erase in Resuming an Erase Flash Sector Operation
- Add list of specific commands impacted by Flash Access Protection
- Clarify writability of ACCERR and FPVIOL while CCIF is set in FSTAT register description

- Changed references of 64-bit FAC to 32-bit FAC in the register and field descriptions of the following: Execute-only Access Registers (FTFA_XACCn), Supervisor-only Access Registers (FTFA_SACCn), Flash Access Segment Size Register (FTFA_FACSS) and Flash Access Segment Number Register (FTFA_FACSN).

44.33 CRC changes

- No substantial content changes

44.34 ADC changes

- No substantial content changes

44.35 CMP changes

- No substantial content changes

44.36 DAC changes

- No substantial content changes

44.37 FTM changes

- For section "Counter clock source", changed "SC register selects clock sources" to "SC register selects one of three possible clock sources"

- Made the following changes to section Channel trigger output to clarify external trigger functionality:
  - Updated figure "Channel match trigger"
  - Updated the first two paragraphs to "If CH(j)TRIG bit of the FTM External Trigger (FTM_EXTTRIG) register is set, where j = 0, 1, 2, 3, 4, or 5, then the FTM generates a trigger when the channel (j) match occurs (FTM counter = C(j)V). The channel trigger output provides a trigger signal which has one FTM clock period width and is used for on-chip modules."
- In FTM_EXTTRIG register description, added cross-references to Channel trigger output and Initialization trigger sections
- Updated description of FTM_SC[CLKS] field
- In section Edge-Aligned PWM (EPWM) mode, updated sentence to read as follows: "If (CnV > MOD), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match."

- For section "Modes of operation" and section "Counter clock source", changed "MCU" to "chip"
- For section "Counter clock source", changed "Refer to the chip specific documentation for further information." to "see the chip-specific FTM information for further details."

- In section Filter for Input Capture mode :
  - Added more details about input capture delay
  - Added figure "Input capture example"
  - Corrected "Channel input filter example" figure

- In section FlexTimer philosophy, updated the first sentence to "The FlexTimer is built upon a simple timer, the Timer PWM Module – TPM, used for many years on our HCS08 family of 8-bit microcontrollers."

44.38 LPTMR changes

- No substantial content changes

44.39 PDB changes

- In PDB trigger input source selection section, the sentence "SC[TRIGSEL] selects the active trigger input source or software trigger." was removed.

- In "Pulse-Out's" section, added a Pulse-Out generation diagram.
- Updated LDOK with a slightly more clear description.
- Added a note about SC[LDOK] to Modulus register (PDB_MOD), Interrupt Delay register (PDB_IDLY), PODLY, DACINTx, and registers.
- Deleted “PDB signal description” section.

- In Pulse-Out n Enable register (POEN), in bit POEN (PDB Pulse-Out Enable), changed "Enables the pulse output. Only lower Y bits are implemented in this MCU." to "Enables the pulse output. Only lower 8 bits are implemented in this MCU."

- In Status and Control register (PDB_SC) register, updated Load Mode Select (LDMOD) field description; made PRESCALAR field more clear; updated PDB Interrupt Flag (PDBIF) field description.
- In Channel n Status register (PDB_CHnS) register, updated PDB Channel Flags (CF) field description.
- In "PDB pre-trigger and trigger outputs" section, replaced "When the PDB counter reaches the value set in IDLY register," with "When the PDB counter reaches the value (CNT + 1),"
- In "Updating the delay registers" section, "Circumstances of update to the delay registers" table was renamed to "When delay registers are updated" table; in that table, SC[LDMOD] rows 01 and 11 were updated using "The PDB counter reaches PDB_MOD[MOD] + 1 value".

- In Status and Control register field SC[LDMOD], for 01 value: "The internal registers are loaded with the values from their buffers when the PDB counter reaches the MOD register value after 1 is written to LDOK." is changed to "The internal registers are loaded with the values from their buffers when the PDB counter (CNT) =MOD + 1 CNT delay elapsed, after 1 is written to LDOK."
- In Status and Control register field SC[LDMOD], for 11 value: "The internal registers are loaded with the values from their buffers when either the PDB counter reaches the MOD register value or a trigger input event is detected, after 1 is written to LDOK." is changed to "The internal registers are loaded with the values from their buffers when either the PDB counter (CNT) = MOD + 1 CNT delay elapsed, or a trigger input event is detected, after 1 is written to LDOK.".
- In Channel n Status register field PDB_CHnS[CF] (the PDB Channel Flags): "The CF[m] field is set when the PDB counter matches the CHnDLYm. Write 0 to clear these bits." is changed to "The CF[m] field is set when the PDB counter (PDB_CNT) matches the value CHnDLYm + 1. Write 0 to clear CF."
- In Status and Control Register (SC) Load OK field (LDOK, bit0): "After 1 is written to the LDOK field, the values in the internal buffers of these registers are not effective, and new values cannot be written to the internal buffers until the existing values in the internal buffers are loaded into their corresponding registers." is changed to "Before 1 is written to the LDOK field, the values in the internal buffers of these registers are not effective, and new values cannot be written to the internal buffers until the existing values in the internal buffers are loaded into their corresponding registers."
- In Counter Register (CNT) description,added a note: "Writing to this read-only register will generate a transfer error (and possibly a hard fault)."

44.40 SPI module changes

• Changed "MCU" to "chip" throughout the chapter.
• Removed note from Section, SIN—Serial Input
• In , Bit 14 changed from ROZ to RXCTR4; Bit 11 changed from ROZ to TXCTR4. Updated bit field width for CMDCTR from 4 to 5 bits.
• Added note to SPI_SR[TFFF] bit field.
• In Section, Modified SPI Transfer Format (MTFE = 1, CPHA = 1), added note, "When using MTFE=1...POP operation."
• In Modified SPI Transfer Format (MTFE = 1, CPHA = 1) - Updated note.
• In Status Register (SPI_SR) - Changed TXRXS bitfield access from W1C to RO.
• In Fast Continuous Selection Format, updated note. • Editorial updates

• Deleted note from SOUT—Serial Output.
• Clarified Memory Map/Register Definition introductory text by changing second sentence:
  • From: "Write access to the POPR and RXFRn also results in a transfer error."
  • To: "Any Write access to the POPR and RXFRn also results in a transfer error."

• Updated PUSH TX FIFO Register In Master Mode (SPI_PUSHR) description.
• In section, PCS0/$\overline{SS}$—Peripheral Chip Select/Slave Select : Added note.
• In Transmit FIFO Fill Interrupt or DMA Request added text to Note: "Configure the DMA to fill only one FIFO location per transfer." • In Receive FIFO Drain Interrupt or DMA Request added text: "Configure the DMA to drain only one FIFO location per transfer."
• In PUSH TX FIFO Register In Master Mode (SPI_PUSHR) description, changed second sentence from "Only 16-bits can be written into TXDATA field" to "User must write 16-bits data into TXDATA field."
• Updated bit field descriptions for SR[TFFF, RFDF]

• In Status Register (SPI_SR), edited field description of:
  • RFDF
  • TFFF

• In MTFE, added note When MTFE=1 with continuous SCK enabled.. • In Modified SPI Transfer Format (MTFE = 1, CPHA = 1) removed note When MTFE=1 with continuous SCK enabled..

44.41 I2C changes

• No substantial content changes

44.42 FlexCAN module changes

• In section "Freeze mode", "CAN_CTRL1[CLK_SRC]" corrected to "CAN_CTRL1[CLKSRC]".
• In FlexCAN memory mapping, changed reference to address offset in second paragraph. • In Table 39-2, prefixed "CAN_" to all register names.
• Changed "MCU" to "chip" throughout the chapter.

• FlexCAN module features   • Changed 64 to 16 in bullet "Flexible message buffers (MBs), totaling..."
• Module Configuration Register (CAN_MCR)   • Removed unnecessary text in CAN_MCR[FRZ] and CAN_MCR[HALT] field descriptions.
• In Message buffer structure, removed note about configuring CAN FD frames.
• Changed access type and description for reserved field at bit location 15 in CAN_CTRL2.
• In the introduction section of Memory map/register definition, added text explaining that there is a separate message buffer memory map section, and added a link to that section.
• Table in CAN_CTRL2[RFFN] field description was modified to show only information for 16 message buffers.
• In Protocol timing, in bullet point about Time Segment 1 changed "4 to 16 time quanta" to "2 to 16 time quanta." • In Figure 39-3, changed "4...16" to "2...16" for Time Segment 1.
• In Module Configuration Register (CAN_MCR), changed access type of MCR[11] from RW to ROZ. • In Module Configuration Register (CAN_MCR), changed access type of MCR[14] from RU to ROZ. • In Error Counter (CAN_ECR), changed access type of ECR[24-31] from RW to ROZ. • In Error Counter (CAN_ECR), changed access type of ECR[16-23] from RW to ROZ. • In Control 2 register (CAN_CTRL2), changed access type of CTRL2[31] from RW to ROZ.
• In , made field MBDSR1 reserved.
• Modified description of Rx Individual Mask Registers (CAN_RXIMR*n*) to note option of using RXIMR memory region as general access memory. • Modified Bus interface to explain how to use RXIMR memory region as general access memory.

44.43 UART changes

• Added an overbar over the CTS signal name in Table 42-1.

44.44 GPIO changes

• No substantial content changes

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number KV11P64M75RM
Revision 4, May 2017