

---

# K02F Sub-Family Reference Manual

Supports: MK02FN128VLH10, MK02FN128VLF10,  
MK02FN128VFM10, MK02FN64VLH10, MK02FN64VLF10,  
MK02FN64VFM10

Document Number: K02P64M100SFARM  
Rev. 1, 7/2014





# Contents

Section number	Title	Page
<b>Chapter 1</b>		
<b>About This Document</b>		
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
<b>Chapter 2</b>		
<b>Introduction</b>		
2.1	Overview.....	41
2.2	Module Functional Categories.....	41
2.2.1	ARM® Cortex®-M4 Core Modules.....	42
2.2.2	System Modules.....	43
2.2.3	Memories and Memory Interfaces.....	43
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
2.3	Orderable part numbers.....	47
<b>Chapter 3</b>		
<b>Chip Configuration</b>		
3.1	Introduction.....	49

<b>Section number</b>	<b>Title</b>	<b>Page</b>
3.2	Core modules.....	49
3.2.1	ARM Cortex-M4 Core Configuration.....	49
3.2.2	Nested Vectored Interrupt Controller (NVIC) Configuration.....	51
3.2.3	Asynchronous Wake-up Interrupt Controller (AWIC) Configuration.....	57
3.2.4	FPU Configuration.....	58
3.2.5	JTAG Controller Configuration.....	58
3.3	System modules.....	59
3.3.1	SIM Configuration.....	59
3.3.2	System Mode Controller (SMC) Configuration.....	60
3.3.3	PMC Configuration.....	60
3.3.4	Low-Leakage Wake-up Unit (LLWU) Configuration.....	61
3.3.5	MCM Configuration.....	62
3.3.6	Crossbar-Light Switch Configuration.....	63
3.3.7	Peripheral Bridge Configuration.....	65
3.3.8	DMA request multiplexer configuration.....	66
3.3.9	DMA Controller Configuration.....	68
3.3.10	External Watchdog Monitor (EWM) Configuration.....	69
3.3.11	Watchdog Configuration.....	71
3.4	Clock modules.....	72
3.4.1	MCG Configuration.....	72
3.4.2	OSC Configuration.....	74
3.5	Memories and memory interfaces.....	74
3.5.1	Flash Memory Configuration.....	74
3.5.2	Flash Memory Controller Configuration.....	77
3.5.3	SRAM Configuration.....	78
3.6	Security.....	79
3.6.1	CRC Configuration.....	79

<b>Section number</b>	<b>Title</b>	<b>Page</b>
3.7	Analog.....	80
3.7.1	16-bit SAR ADC Configuration.....	80
3.7.2	CMP Configuration.....	85
3.7.3	12-bit DAC Configuration.....	87
3.7.4	VREF Configuration.....	89
3.8	Timers.....	90
3.8.1	PDB Configuration.....	90
3.8.2	FlexTimer Configuration.....	93
3.8.3	PIT Configuration.....	99
3.8.4	Low-power timer configuration.....	100
3.9	Communication interfaces.....	102
3.9.1	SPI configuration.....	102
3.9.2	I2C Configuration.....	105
3.9.3	UART Configuration.....	106
3.10	Human-machine interfaces.....	108
3.10.1	GPIO configuration.....	108

## **Chapter 4 Memory Map**

4.1	Introduction.....	111
4.2	System memory map.....	111
4.2.1	Aliased bit-band regions.....	112
4.2.2	Flash Access Control Introduction.....	114
4.3	Flash Memory Map.....	114
4.3.1	Alternate Non-Volatile IRC User Trim Description.....	115
4.4	SRAM memory map.....	115
4.5	Peripheral bridge (AIPS-Lite) memory map.....	115
4.5.1	Read-after-write sequence and required serialization of memory operations.....	116
4.5.2	Peripheral Bridge 0 (AIPS-Lite 0) Memory Map.....	116
4.6	Private Peripheral Bus (PPB) memory map.....	120

## Chapter 5 Clock Distribution

5.1	Introduction.....	121
5.2	Programming model.....	121
5.3	High-Level device clocking diagram.....	121
5.4	Clock definitions.....	122
5.4.1	Device clock summary.....	123
5.5	Internal clocking requirements.....	125
5.5.1	Clock divider values after reset.....	126
5.5.2	VLPR mode clocking.....	126
5.6	Clock Gating.....	127
5.7	Module clocks.....	127
5.7.1	PMC 1-kHz LPO clock.....	128
5.7.2	IRC 48MHz clock.....	128
5.7.3	WDOG clocking.....	129
5.7.4	Debug trace clock.....	129
5.7.5	PORT digital filter clocking.....	130
5.7.6	LPTMR clocking.....	130
5.7.7	CLKOUT32K clocking.....	131
5.7.8	UART clocking.....	131

## Chapter 6 Reset and Boot

6.1	Introduction.....	133
6.2	Reset.....	133
6.2.1	Power-on reset (POR).....	134
6.2.2	System reset sources.....	134
6.2.3	MCU Resets.....	137
6.2.4	Reset Pin .....	139

<b>Section number</b>	<b>Title</b>	<b>Page</b>
6.2.5	Debug resets.....	139
6.3	Boot.....	140
6.3.1	Boot sources.....	140
6.3.2	Boot options.....	140
6.3.3	FOPT boot options.....	141
6.3.4	Boot sequence.....	142

## **Chapter 7 Power Management**

7.1	Introduction.....	145
7.2	Clocking modes.....	145
7.2.1	Partial Stop.....	145
7.2.2	DMA Wakeup.....	146
7.2.3	Compute Operation.....	147
7.2.4	Peripheral Doze.....	148
7.2.5	Clock Gating.....	149
7.3	Power Modes Description.....	149
7.4	Entering and exiting power modes.....	151
7.5	Power mode transitions.....	152
7.6	Power modes shutdown sequencing.....	153
7.7	Flash Program Restrictions.....	154
7.8	Module Operation in Low Power Modes.....	154

## **Chapter 8 Security**

8.1	Introduction.....	159
8.2	Flash Security.....	159
8.3	Security Interactions with other Modules.....	160
8.3.1	Security Interactions with Debug.....	160

Section number	Title	Page
<b>Chapter 9</b>		
<b>Debug</b>		
9.1	Introduction.....	161
9.1.1	References.....	163
9.2	The Debug Port.....	164
9.2.1	JTAG-to-SWD change sequence.....	164
9.2.2	JTAG-to-cJTAG change sequence.....	165
9.3	Debug Port Pin Descriptions.....	165
9.4	System TAP connection.....	165
9.4.1	IR Codes.....	166
9.5	JTAG status and control registers.....	166
9.5.1	MDM-AP Control Register.....	167
9.5.2	MDM-AP Status Register.....	169
9.6	Debug Resets.....	170
9.7	AHB-AP.....	171
9.8	ITM.....	171
9.9	Core Trace Connectivity.....	172
9.10	TPIU.....	172
9.11	DWT.....	172
9.12	Debug in Low Power Modes.....	173
9.12.1	Debug Module State in Low Power Modes.....	173
9.13	Debug & Security.....	174

**Chapter 10**  
**Signal Multiplexing and Signal Descriptions**

10.1	Introduction.....	175
10.2	Signal Multiplexing Integration.....	175
10.2.1	Port control and interrupt module features.....	176
10.2.2	Clock gating.....	177
10.2.3	Signal multiplexing constraints.....	177



Section number	Title	Page
10.3	Pinout.....	177
10.3.1	K02F Signal Multiplexing and Pin Assignments.....	177
10.3.2	K02F Pinouts.....	180
10.4	Module Signal Description Tables.....	183
10.4.1	Core Modules.....	183
10.4.2	System Modules.....	184
10.4.3	Clock Modules.....	184
10.4.4	Analog.....	185
10.4.5	Timer Modules.....	186
10.4.6	Communication Interfaces.....	187
10.4.7	Human-Machine Interfaces (HMI).....	188

## Chapter 11 Port Control and Interrupts (PORT)

11.1	Introduction.....	189
11.2	Overview.....	189
11.2.1	Features.....	189
11.2.2	Modes of operation.....	190
11.3	External signal description.....	191
11.4	Detailed signal description.....	191
11.5	Memory map and register definition.....	191
11.5.1	Pin Control Register n (PORTx_PCRn).....	198
11.5.2	Global Pin Control Low Register (PORTx_GPCLR).....	200
11.5.3	Global Pin Control High Register (PORTx_GPCHR).....	201
11.5.4	Interrupt Status Flag Register (PORTx_ISFR).....	202
11.5.5	Digital Filter Enable Register (PORTx_DFER).....	202
11.5.6	Digital Filter Clock Register (PORTx_DFCL).....	203
11.5.7	Digital Filter Width Register (PORTx_DFWR).....	203
11.6	Functional description.....	204
11.6.1	Pin control.....	204

Section number	Title	Page
11.6.2	Global pin control.....	205
11.6.3	External interrupts.....	205
11.6.4	Digital filter.....	206

## Chapter 12 System Integration Module (SIM)

12.1	Introduction.....	209
12.1.1	Features.....	209
12.2	Memory map and register definition.....	210
12.2.1	System Options Register 1 (SIM_SOPT1).....	211
12.2.2	SOPT1 Configuration Register (SIM_SOPT1CFG).....	212
12.2.3	System Options Register 2 (SIM_SOPT2).....	213
12.2.4	System Options Register 4 (SIM_SOPT4).....	215
12.2.5	System Options Register 5 (SIM_SOPT5).....	217
12.2.6	System Options Register 7 (SIM_SOPT7).....	219
12.2.7	System Options Register 8 (SIM_SOPT8).....	220
12.2.8	System Device Identification Register (SIM_SDID).....	222
12.2.9	System Clock Gating Control Register 4 (SIM_SCGC4).....	224
12.2.10	System Clock Gating Control Register 5 (SIM_SCGC5).....	226
12.2.11	System Clock Gating Control Register 6 (SIM_SCGC6).....	227
12.2.12	System Clock Gating Control Register 7 (SIM_SCGC7).....	230
12.2.13	System Clock Divider Register 1 (SIM_CLKDIV1).....	230
12.2.14	Flash Configuration Register 1 (SIM_FCFG1).....	233
12.2.15	Flash Configuration Register 2 (SIM_FCFG2).....	235
12.2.16	Unique Identification Register High (SIM_UIDH).....	235
12.2.17	Unique Identification Register Mid-High (SIM_UIDMH).....	236
12.2.18	Unique Identification Register Mid Low (SIM_UIDML).....	236
12.2.19	Unique Identification Register Low (SIM_UIDL).....	237
12.3	Functional description.....	237

Section number	Title	Page
<b>Chapter 13</b>		
<b>Kinetis Flashloader</b>		
13.1	Chip-Specific Information.....	239
13.2	Introduction.....	239
13.3	Functional Description.....	241
13.3.1	Memory Maps.....	241
13.3.2	Kinetis Flashloader.....	241
13.3.3	Start-up Process.....	242
13.3.4	Clock Configuration.....	242
13.3.5	Flashloader Protocol.....	243
13.3.6	Flashloader Packet Types.....	247
13.3.7	Flashloader Command API.....	254
13.3.8	Flashloader Exit state.....	267
13.4	Peripherals Supported.....	269
13.4.1	I2C Peripheral.....	269
13.4.2	SPI Peripheral.....	270
13.4.3	UART Peripheral.....	272
13.5	Get/SetProperty Command Properties.....	275
13.5.1	Property Definitions.....	276
13.6	Kinetis Flashloader Status Error Codes.....	277

**Chapter 14**  
**Reset Control Module (RCM)**

14.1	Introduction.....	279
14.2	Reset memory map and register descriptions.....	279
14.2.1	System Reset Status Register 0 (RCM_SRS0).....	280
14.2.2	System Reset Status Register 1 (RCM_SRS1).....	281
14.2.3	Reset Pin Filter Control register (RCM_RPFC).....	283
14.2.4	Reset Pin Filter Width register (RCM_RPFW).....	284
14.2.5	Sticky System Reset Status Register 0 (RCM_SSRS0).....	285

Section number	Title	Page
14.2.6	Sticky System Reset Status Register 1 (RCM_SSRS1).....	286

## Chapter 15 System Mode Controller (SMC)

15.1	Introduction.....	289
15.2	Modes of operation.....	289
15.3	Memory map and register descriptions.....	291
15.3.1	Power Mode Protection register (SMC_PMPROT).....	292
15.3.2	Power Mode Control register (SMC_PMCTRL).....	293
15.3.3	Stop Control Register (SMC_STOPCTRL).....	295
15.3.4	Power Mode Status register (SMC_PMSTAT).....	296
15.4	Functional description.....	297
15.4.1	Power mode transitions.....	297
15.4.2	Power mode entry/exit sequencing.....	300
15.4.3	Run modes.....	302
15.4.4	Wait modes.....	304
15.4.5	Stop modes.....	305
15.4.6	Debug in low power modes.....	308

## Chapter 16 Power Management Controller (PMC)

16.1	Introduction.....	311
16.2	Features.....	311
16.3	Low-voltage detect (LVD) system.....	311
16.3.1	LVD reset operation.....	312
16.3.2	LVD interrupt operation.....	312
16.3.3	Low-voltage warning (LVW) interrupt operation.....	312
16.4	I/O retention.....	313
16.5	Memory map and register descriptions.....	313
16.5.1	Low Voltage Detect Status And Control 1 register (PMC_LVDSC1).....	314
16.5.2	Low Voltage Detect Status And Control 2 register (PMC_LVDSC2).....	315

Section number	Title	Page
16.5.3	Regulator Status And Control register (PMC_REGSC).....	316

## Chapter 17 Low-Leakage Wakeup Unit (LLWU)

17.1	Introduction.....	319
17.1.1	Features.....	319
17.1.2	Modes of operation.....	320
17.1.3	Block diagram.....	321
17.2	LLWU signal descriptions.....	322
17.3	Memory map/register definition.....	322
17.3.1	LLWU Pin Enable 1 register (LLWU_PE1).....	323
17.3.2	LLWU Pin Enable 2 register (LLWU_PE2).....	324
17.3.3	LLWU Pin Enable 3 register (LLWU_PE3).....	325
17.3.4	LLWU Pin Enable 4 register (LLWU_PE4).....	326
17.3.5	LLWU Module Enable register (LLWU_ME).....	327
17.3.6	LLWU Flag 1 register (LLWU_F1).....	329
17.3.7	LLWU Flag 2 register (LLWU_F2).....	331
17.3.8	LLWU Flag 3 register (LLWU_F3).....	332
17.3.9	LLWU Pin Filter 1 register (LLWU_FILT1).....	334
17.3.10	LLWU Pin Filter 2 register (LLWU_FILT2).....	335
17.4	Functional description.....	336
17.4.1	LLS mode.....	337
17.4.2	VLLS modes.....	337
17.4.3	Initialization.....	337

## Chapter 18 Miscellaneous Control Module (MCM)

18.1	Introduction.....	339
18.1.1	Features.....	339
18.2	Memory map/register descriptions.....	339
18.2.1	Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC).....	340

Section number	Title	Page
18.2.2	Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC).....	340
18.2.3	Crossbar Switch (AXBS) Control Register (MCM_PLACR).....	341
18.2.4	Interrupt Status and Control Register (MCM_ISCR).....	342
18.2.5	Compute Operation Control Register (MCM_CPO).....	345
18.3	Functional description.....	346
18.3.1	Interrupts.....	346

## Chapter 19 Crossbar Switch Lite (AXBS-Lite)

19.1	Introduction.....	347
19.1.1	Features.....	347
19.2	Memory Map / Register Definition.....	348
19.3	Functional Description.....	348
19.3.1	General operation.....	348
19.3.2	Arbitration.....	349
19.4	Initialization/application information.....	350

## Chapter 20 Peripheral Bridge (AIPS-Lite)

20.1	Introduction.....	353
20.1.1	Features.....	353
20.1.2	General operation.....	353
20.2	Functional description.....	354
20.2.1	Access support.....	354

## Chapter 21 Direct Memory Access Multiplexer (DMAMUX)

21.1	Introduction.....	355
21.1.1	Overview.....	355
21.1.2	Features.....	356
21.1.3	Modes of operation.....	356
21.2	External signal description.....	357

Section number	Title	Page
21.3	Memory map/register definition.....	357
21.3.1	Channel Configuration register (DMAMUX_CHCFGn).....	358
21.4	Functional description.....	358
21.4.1	DMA channels with periodic triggering capability.....	359
21.4.2	DMA channels with no triggering capability.....	361
21.4.3	Always-enabled DMA sources.....	361
21.5	Initialization/application information.....	363
21.5.1	Reset.....	363
21.5.2	Enabling and configuring sources.....	363

## Chapter 22 Enhanced Direct Memory Access (eDMA)

22.1	Introduction.....	367
22.1.1	eDMA system block diagram.....	367
22.1.2	Block parts.....	368
22.1.3	Features.....	369
22.2	Modes of operation.....	371
22.3	Memory map/register definition.....	371
22.3.1	TCD memory.....	371
22.3.2	TCD initialization.....	372
22.3.3	TCD structure.....	372
22.3.4	Reserved memory and bit fields.....	374
22.3.1	Control Register (DMA_CR).....	377
22.3.2	Error Status Register (DMA_ES).....	380
22.3.3	Enable Request Register (DMA_ERQ).....	382
22.3.4	Enable Error Interrupt Register (DMA_EEI).....	383
22.3.5	Clear Enable Error Interrupt Register (DMA_CEEI).....	384
22.3.6	Set Enable Error Interrupt Register (DMA_SEEI).....	385
22.3.7	Clear Enable Request Register (DMA_CERQ).....	386
22.3.8	Set Enable Request Register (DMA_SERQ).....	387

Section number	Title	Page
22.3.9	Clear DONE Status Bit Register (DMA_CDNE).....	388
22.3.10	Set START Bit Register (DMA_SSRT).....	389
22.3.11	Clear Error Register (DMA_CERR).....	390
22.3.12	Clear Interrupt Request Register (DMA_CINT).....	391
22.3.13	Interrupt Request Register (DMA_INT).....	392
22.3.14	Error Register (DMA_ERR).....	393
22.3.15	Hardware Request Status Register (DMA_HRS).....	394
22.3.16	Enable Asynchronous Request in Stop Register (DMA_EARS).....	396
22.3.17	Channel n Priority Register (DMA_DCHPRI <sub>n</sub> ).....	397
22.3.18	TCD Source Address (DMA_TCD <sub>n</sub> _SADDR).....	398
22.3.19	TCD Signed Source Address Offset (DMA_TCD <sub>n</sub> _SOFF).....	398
22.3.20	TCD Transfer Attributes (DMA_TCD <sub>n</sub> _ATTR).....	399
22.3.21	TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD <sub>n</sub> _NBYTES_MLNO).....	400
22.3.22	TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD <sub>n</sub> _NBYTES_MLOFFNO).....	400
22.3.23	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD <sub>n</sub> _NBYTES_MLOFFYES).....	402
22.3.24	TCD Last Source Address Adjustment (DMA_TCD <sub>n</sub> _SLAST).....	403
22.3.25	TCD Destination Address (DMA_TCD <sub>n</sub> _DADDR).....	403
22.3.26	TCD Signed Destination Address Offset (DMA_TCD <sub>n</sub> _DOFF).....	404
22.3.27	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD <sub>n</sub> _CITER_ELINKYES).....	404
22.3.28	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD <sub>n</sub> _CITER_ELINKNO).....	406
22.3.29	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD <sub>n</sub> _DLASTSGA).....	407
22.3.30	TCD Control and Status (DMA_TCD <sub>n</sub> _CSR).....	407
22.3.31	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD <sub>n</sub> _BITER_ELINKYES).....	410
22.3.32	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD <sub>n</sub> _BITER_ELINKNO).....	411



Section number	Title	Page
22.4	Functional description.....	412
22.4.1	eDMA basic data flow.....	412
22.4.2	Fault reporting and handling.....	415
22.4.3	Channel preemption.....	416
22.4.4	Performance.....	417
22.5	Initialization/application information.....	421
22.5.1	eDMA initialization.....	421
22.5.2	Programming errors.....	423
22.5.3	Arbitration mode considerations.....	424
22.5.4	Performing DMA transfers.....	424
22.5.5	Monitoring transfer descriptor status.....	428
22.5.6	Channel Linking.....	430
22.5.7	Dynamic programming.....	431

## Chapter 23 External Watchdog Monitor (EWM)

23.1	Introduction.....	437
23.1.1	Features.....	437
23.1.2	Modes of Operation.....	438
23.1.3	Block Diagram.....	439
23.2	EWM Signal Descriptions.....	440
23.3	Memory Map/Register Definition.....	440
23.3.1	Control Register (EWM_CTRL).....	440
23.3.2	Service Register (EWM_SERV).....	441
23.3.3	Compare Low Register (EWM_CMPL).....	441
23.3.4	Compare High Register (EWM_CMPH).....	442
23.3.5	Clock Prescaler Register (EWM_CLKPRESCALER).....	443
23.4	Functional Description.....	443
23.4.1	The EWM_out Signal.....	443
23.4.2	The EWM_in Signal.....	444

Section number	Title	Page
23.4.3	EWM Counter.....	445
23.4.4	EWM Compare Registers.....	445
23.4.5	EWM Refresh Mechanism.....	445
23.4.6	EWM Interrupt.....	446
23.4.7	Counter clock prescaler.....	446

## Chapter 24 Watchdog Timer (WDOG)

24.1	Introduction.....	447
24.2	Features.....	447
24.3	Functional overview.....	449
24.3.1	Unlocking and updating the watchdog.....	450
24.3.2	Watchdog configuration time (WCT).....	451
24.3.3	Refreshing the watchdog.....	452
24.3.4	Windowed mode of operation.....	452
24.3.5	Watchdog disabled mode of operation.....	452
24.3.6	Debug modes of operation.....	453
24.4	Testing the watchdog.....	453
24.4.1	Quick test.....	454
24.4.2	Byte test.....	454
24.5	Backup reset generator.....	455
24.6	Generated resets and interrupts.....	456
24.7	Memory map and register definition.....	456
24.7.1	Watchdog Status and Control Register High (WDOG_STCTRLH).....	457
24.7.2	Watchdog Status and Control Register Low (WDOG_STCTRL).....	459
24.7.3	Watchdog Time-out Value Register High (WDOG_TOVALH).....	459
24.7.4	Watchdog Time-out Value Register Low (WDOG_TOVAL).....	460
24.7.5	Watchdog Window Register High (WDOG_WINH).....	460
24.7.6	Watchdog Window Register Low (WDOG_WINL).....	461
24.7.7	Watchdog Refresh register (WDOG_REFRESH).....	461

Section number	Title	Page
24.7.8	Watchdog Unlock register (WDOG_UNLOCK).....	461
24.7.9	Watchdog Timer Output Register High (WDOG_TMROUTH).....	462
24.7.10	Watchdog Timer Output Register Low (WDOG_TMROUTL).....	462
24.7.11	Watchdog Reset Count register (WDOG_RSTCNT).....	463
24.7.12	Watchdog Prescaler register (WDOG_PRESC).....	463
24.8	Watchdog operation with 8-bit access.....	463
24.8.1	General guideline.....	463
24.8.2	Refresh and unlock operations with 8-bit access.....	464
24.9	Restrictions on watchdog operation.....	465

## Chapter 25 Multipurpose Clock Generator (MCG)

25.1	Introduction.....	467
25.1.1	Features.....	467
25.1.2	Modes of Operation.....	469
25.2	External Signal Description.....	470
25.3	Memory Map/Register Definition.....	470
25.3.1	MCG Control 1 Register (MCG_C1).....	470
25.3.2	MCG Control 2 Register (MCG_C2).....	472
25.3.3	MCG Control 3 Register (MCG_C3).....	473
25.3.4	MCG Control 4 Register (MCG_C4).....	474
25.3.5	MCG Control 6 Register (MCG_C6).....	475
25.3.6	MCG Status Register (MCG_S).....	476
25.3.7	MCG Status and Control Register (MCG_SC).....	477
25.3.8	MCG Auto Trim Compare Value High Register (MCG_ATCVH).....	478
25.3.9	MCG Auto Trim Compare Value Low Register (MCG_ATCVL).....	478
25.3.10	MCG Control 7 Register (MCG_C7).....	479
25.3.11	MCG Control 8 Register (MCG_C8).....	479

Section number	Title	Page
25.4	Functional description.....	480
25.4.1	MCG mode state diagram.....	480
25.4.2	Low-power bit usage.....	484
25.4.3	MCG Internal Reference Clocks.....	484
25.4.4	External Reference Clock.....	484
25.4.5	MCG Fixed Frequency Clock .....	485
25.4.6	MCG Auto TRIM (ATM).....	485
25.5	Initialization / Application information.....	487
25.5.1	MCG module initialization sequence.....	487
25.5.2	Using a 32.768 kHz reference.....	489
25.5.3	MCG mode switching.....	490

## Chapter 26 Oscillator (OSC)

26.1	Introduction.....	497
26.2	Features and Modes.....	497
26.3	Block Diagram.....	498
26.4	OSC Signal Descriptions.....	498
26.5	External Crystal / Resonator Connections.....	499
26.6	External Clock Connections.....	500
26.7	Memory Map/Register Definitions.....	501
26.7.1	OSC Memory Map/Register Definition.....	501
26.8	Functional Description.....	503
26.8.1	OSC module states.....	503
26.8.2	OSC module modes.....	505
26.8.3	Counter.....	507
26.8.4	Reference clock pin requirements.....	507
26.9	Reset.....	507
26.10	Low power modes operation.....	508
26.11	Interrupts.....	508

Section number	Title	Page
<b>Chapter 27</b>		
<b>Flash Memory Controller (FMC)</b>		
27.1	Introduction.....	509
27.1.1	Overview.....	509
27.1.2	Features.....	509
27.2	Modes of operation.....	510
27.3	External signal description.....	510
27.4	Memory map and register descriptions.....	510
27.4.1	Flash Access Protection Register (FMC_PFAPR).....	515
27.4.2	Flash Bank 0 Control Register (FMC_PFB0CR).....	519
27.4.3	Flash Bank 1 Control Register (FMC_PFB1CR).....	522
27.4.4	Cache Tag Storage (FMC_TAGVDW0Sn).....	524
27.4.5	Cache Tag Storage (FMC_TAGVDW1Sn).....	525
27.4.6	Cache Tag Storage (FMC_TAGVDW2Sn).....	526
27.4.7	Cache Tag Storage (FMC_TAGVDW3Sn).....	527
27.4.8	Cache Data Storage (upper word) (FMC_DATAW0SnU).....	527
27.4.9	Cache Data Storage (lower word) (FMC_DATAW0SnL).....	528
27.4.10	Cache Data Storage (upper word) (FMC_DATAW1SnU).....	528
27.4.11	Cache Data Storage (lower word) (FMC_DATAW1SnL).....	529
27.4.12	Cache Data Storage (upper word) (FMC_DATAW2SnU).....	529
27.4.13	Cache Data Storage (lower word) (FMC_DATAW2SnL).....	530
27.4.14	Cache Data Storage (upper word) (FMC_DATAW3SnU).....	530
27.4.15	Cache Data Storage (lower word) (FMC_DATAW3SnL).....	531
27.5	Functional description.....	531
27.5.1	Default configuration.....	531
27.5.2	Configuration options.....	532
27.5.3	Speculative reads.....	532
27.6	Initialization and application information.....	533

## Chapter 28 Flash Memory Module (FTFA)

28.1	Introduction.....	535
28.1.1	Features.....	536
28.1.2	Block Diagram.....	536
28.1.3	Glossary.....	537
28.2	External Signal Description.....	538
28.3	Memory Map and Registers.....	538
28.3.1	Flash Configuration Field Description.....	539
28.3.2	Program Flash IFR Map.....	539
28.3.3	Register Descriptions.....	540
28.4	Functional Description.....	554
28.4.1	Flash Protection.....	554
28.4.2	Flash Access Protection.....	555
28.4.3	Interrupts.....	556
28.4.4	Flash Operation in Low-Power Modes.....	557
28.4.5	Functional Modes of Operation.....	557
28.4.6	Flash Reads and Ignored Writes.....	557
28.4.7	Read While Write (RWW).....	558
28.4.8	Flash Program and Erase.....	558
28.4.9	Flash Command Operations.....	558
28.4.10	Margin Read Commands.....	562
28.4.11	Flash Command Description.....	563
28.4.12	Security.....	577
28.4.13	Reset Sequence.....	579

## Chapter 29 Cyclic Redundancy Check (CRC)

29.1	Introduction.....	581
29.1.1	Features.....	581

Section number	Title	Page
29.1.2	Block diagram.....	581
29.1.3	Modes of operation.....	582
29.2	Memory map and register descriptions.....	582
29.2.1	CRC Data register (CRC_DATA).....	583
29.2.2	CRC Polynomial register (CRC_GPOLY).....	584
29.2.3	CRC Control register (CRC_CTRL).....	584
29.3	Functional description.....	585
29.3.1	CRC initialization/reinitialization.....	585
29.3.2	CRC calculations.....	586
29.3.3	Transpose feature.....	587
29.3.4	CRC result complement.....	589

## Chapter 30 Analog-to-Digital Converter (ADC)

30.1	Introduction.....	591
30.1.1	Features.....	591
30.1.2	Block diagram.....	592
30.2	ADC signal descriptions.....	593
30.2.1	Analog Power (VDDA).....	594
30.2.2	Analog Ground (VSSA).....	594
30.2.3	Voltage Reference Select.....	594
30.2.4	Analog Channel Inputs (ADx).....	595
30.2.5	Differential Analog Channel Inputs (DADx).....	595
30.3	Memory map and register definitions.....	595
30.3.1	ADC Status and Control Registers 1 (ADCx_SC1n).....	596
30.3.2	ADC Configuration Register 1 (ADCx_CFG1).....	599
30.3.3	ADC Configuration Register 2 (ADCx_CFG2).....	601
30.3.4	ADC Data Result Register (ADCx_Rn).....	602
30.3.5	Compare Value Registers (ADCx_CVn).....	603
30.3.6	Status and Control Register 2 (ADCx_SC2).....	604

Section number	Title	Page
30.3.7	Status and Control Register 3 (ADCx_SC3).....	606
30.3.8	ADC Offset Correction Register (ADCx_OFS).....	608
30.3.9	ADC Plus-Side Gain Register (ADCx_PG).....	608
30.3.10	ADC Minus-Side Gain Register (ADCx_MG).....	609
30.3.11	ADC Plus-Side General Calibration Value Register (ADCx_CLPD).....	609
30.3.12	ADC Plus-Side General Calibration Value Register (ADCx_CLPS).....	610
30.3.13	ADC Plus-Side General Calibration Value Register (ADCx_CLP4).....	610
30.3.14	ADC Plus-Side General Calibration Value Register (ADCx_CLP3).....	611
30.3.15	ADC Plus-Side General Calibration Value Register (ADCx_CLP2).....	611
30.3.16	ADC Plus-Side General Calibration Value Register (ADCx_CLP1).....	612
30.3.17	ADC Plus-Side General Calibration Value Register (ADCx_CLP0).....	612
30.3.18	ADC Minus-Side General Calibration Value Register (ADCx_CLMD).....	613
30.3.19	ADC Minus-Side General Calibration Value Register (ADCx_CLMS).....	613
30.3.20	ADC Minus-Side General Calibration Value Register (ADCx_CLM4).....	614
30.3.21	ADC Minus-Side General Calibration Value Register (ADCx_CLM3).....	614
30.3.22	ADC Minus-Side General Calibration Value Register (ADCx_CLM2).....	615
30.3.23	ADC Minus-Side General Calibration Value Register (ADCx_CLM1).....	615
30.3.24	ADC Minus-Side General Calibration Value Register (ADCx_CLM0).....	616
30.4	Functional description.....	616
30.4.1	Clock select and divide control.....	617
30.4.2	Voltage reference selection.....	618
30.4.3	Hardware trigger and channel selects.....	618
30.4.4	Conversion control.....	619
30.4.5	Automatic compare function.....	627
30.4.6	Calibration function.....	628
30.4.7	User-defined offset function.....	630
30.4.8	Temperature sensor.....	631
30.4.9	MCU wait mode operation.....	632



<b>Section number</b>	<b>Title</b>	<b>Page</b>
30.4.10	MCU Normal Stop mode operation.....	632
30.4.11	MCU Low-Power Stop mode operation.....	633
30.5	Initialization information.....	634
30.5.1	ADC module initialization example.....	634
30.6	Application information.....	636
30.6.1	External pins and routing.....	636
30.6.2	Sources of error.....	638

## **Chapter 31 Comparator (CMP)**

31.1	Introduction.....	643
31.1.1	CMP features.....	643
31.1.2	6-bit DAC key features.....	644
31.1.3	ANMUX key features.....	644
31.1.4	CMP, DAC and ANMUX diagram.....	645
31.1.5	CMP block diagram.....	646
31.2	Memory map/register definitions.....	648
31.2.1	CMP Control Register 0 (CMPx_CR0).....	648
31.2.2	CMP Control Register 1 (CMPx_CR1).....	649
31.2.3	CMP Filter Period Register (CMPx_FPR).....	651
31.2.4	CMP Status and Control Register (CMPx_SCR).....	651
31.2.5	DAC Control Register (CMPx_DACCR).....	652
31.2.6	MUX Control Register (CMPx_MUXCR).....	653
31.3	Functional description.....	654
31.3.1	CMP functional modes.....	654
31.3.2	Power modes.....	663
31.3.3	Startup and operation.....	664
31.3.4	Low-pass filter.....	665
31.4	CMP interrupts.....	667
31.5	DMA support.....	667

Section number	Title	Page
31.6	CMP Asynchronous DMA support.....	668
31.7	Digital-to-analog converter.....	669
31.8	DAC functional description.....	669
31.8.1	Voltage reference source select.....	669
31.9	DAC resets.....	670
31.10	DAC clocks.....	670
31.11	DAC interrupts.....	670

## Chapter 32 12-bit Digital-to-Analog Converter (DAC)

32.1	Introduction.....	671
32.2	Features.....	671
32.3	Block diagram.....	671
32.4	Memory map/register definition.....	672
32.4.1	DAC Data Low Register (DACx_DATnL).....	674
32.4.2	DAC Data High Register (DACx_DATnH).....	674
32.4.3	DAC Status Register (DACx_SR).....	675
32.4.4	DAC Control Register (DACx_C0).....	676
32.4.5	DAC Control Register 1 (DACx_C1).....	677
32.4.6	DAC Control Register 2 (DACx_C2).....	678
32.5	Functional description.....	678
32.5.1	DAC data buffer operation.....	678
32.5.2	DMA operation.....	680
32.5.3	Resets.....	680
32.5.4	Low-Power mode operation.....	680

## Chapter 33 Voltage Reference (VREFV1)

33.1	Introduction.....	683
33.1.1	Overview.....	684
33.1.2	Features.....	684

Section number	Title	Page
33.1.3	Modes of Operation.....	685
33.1.4	VREF Signal Descriptions.....	685
33.2	Memory Map and Register Definition.....	686
33.2.1	VREF Trim Register (VREF_TRM).....	686
33.2.2	VREF Status and Control Register (VREF_SC).....	687
33.3	Functional Description.....	688
33.3.1	Voltage Reference Disabled, SC[VREFEN] = 0.....	689
33.3.2	Voltage Reference Enabled, SC[VREFEN] = 1.....	689
33.3.3	Internal voltage regulator.....	690
33.4	Initialization/Application Information.....	691

## Chapter 34 Programmable Delay Block (PDB)

34.1	Introduction.....	693
34.1.1	Features.....	693
34.1.2	Implementation.....	694
34.1.3	Back-to-back acknowledgment connections.....	695
34.1.4	DAC External Trigger Input Connections.....	695
34.1.5	Block diagram.....	695
34.1.6	Modes of operation.....	697
34.2	PDB signal descriptions.....	697
34.3	Memory map and register definition.....	697
34.3.1	Status and Control register (PDBx_SC).....	699
34.3.2	Modulus register (PDBx_MOD).....	701
34.3.3	Counter register (PDBx_CNT).....	702
34.3.4	Interrupt Delay register (PDBx_IDLY).....	702
34.3.5	Channel n Control register 1 (PDBx_CHnC1).....	703
34.3.6	Channel n Status register (PDBx_CHnS).....	704
34.3.7	Channel n Delay 0 register (PDBx_CHnDLY0).....	704
34.3.8	Channel n Delay 1 register (PDBx_CHnDLY1).....	705

Section number	Title	Page
34.3.9	DAC Interval Trigger n Control register (PDBx_DACINTCn).....	705
34.3.10	DAC Interval n register (PDBx_DACINTn).....	706
34.3.11	Pulse-Out n Enable register (PDBx_POEN).....	706
34.3.12	Pulse-Out n Delay register (PDBx_PO $n$ DLY).....	707
34.4	Functional description.....	707
34.4.1	PDB pre-trigger and trigger outputs.....	707
34.4.2	PDB trigger input source selection.....	709
34.4.3	Pulse-Out's.....	709
34.4.4	Updating the delay registers.....	710
34.4.5	Interrupts.....	711
34.4.6	DMA.....	711
34.5	Application information.....	712
34.5.1	Impact of using the prescaler and multiplication factor on timing resolution.....	712

## Chapter 35 FlexTimer Module (FTM)

35.1	Introduction.....	713
35.1.1	FlexTimer philosophy.....	713
35.1.2	Features.....	714
35.1.3	Modes of operation.....	715
35.1.4	Block diagram.....	716
35.2	FTM signal descriptions.....	718
35.3	Memory map and register definition.....	718
35.3.1	Memory map.....	718
35.3.2	Register descriptions.....	719
35.3.3	Status And Control (FTMx_SC).....	724
35.3.4	Counter (FTMx_CNT).....	725
35.3.5	Modulo (FTMx_MOD).....	726
35.3.6	Channel (n) Status And Control (FTMx_CnSC).....	727
35.3.7	Channel (n) Value (FTMx_CnV).....	730

Section number	Title	Page
35.3.8	Counter Initial Value (FTMx_CNTIN).....	730
35.3.9	Capture And Compare Status (FTMx_STATUS).....	731
35.3.10	Features Mode Selection (FTMx_MODE).....	733
35.3.11	Synchronization (FTMx_SYNC).....	735
35.3.12	Initial State For Channels Output (FTMx_OUTINIT).....	737
35.3.13	Output Mask (FTMx_OUTMASK).....	738
35.3.14	Function For Linked Channels (FTMx_COMBINE).....	740
35.3.15	Deadtime Insertion Control (FTMx_DEADTIME).....	745
35.3.16	FTM External Trigger (FTMx_EXTTRIG).....	746
35.3.17	Channels Polarity (FTMx_POL).....	748
35.3.18	Fault Mode Status (FTMx_FMS).....	750
35.3.19	Input Capture Filter Control (FTMx_FILTER).....	752
35.3.20	Fault Control (FTMx_FLTCTRL).....	753
35.3.21	Quadrature Decoder Control And Status (FTMx_QDCTRL).....	756
35.3.22	Configuration (FTMx_CONF).....	758
35.3.23	FTM Fault Input Polarity (FTMx_FLTPOL).....	759
35.3.24	Synchronization Configuration (FTMx_SYNCONF).....	760
35.3.25	FTM Inverting Control (FTMx_INVCTRL).....	762
35.3.26	FTM Software Output Control (FTMx_SWOCTRL).....	763
35.3.27	FTM PWM Load (FTMx_PWMLOAD).....	766
35.4	Functional description.....	767
35.4.1	Clock source.....	768
35.4.2	Prescaler.....	769
35.4.3	Counter.....	769
35.4.4	Input Capture mode.....	775
35.4.5	Output Compare mode.....	778
35.4.6	Edge-Aligned PWM (EPWM) mode.....	779
35.4.7	Center-Aligned PWM (CPWM) mode.....	781

<b>Section number</b>	<b>Title</b>	<b>Page</b>
35.4.8	Combine mode.....	783
35.4.9	Complementary mode.....	790
35.4.10	Registers updated from write buffers.....	791
35.4.11	PWM synchronization.....	793
35.4.12	Inverting.....	809
35.4.13	Software output control.....	810
35.4.14	Deadtime insertion.....	812
35.4.15	Output mask.....	815
35.4.16	Fault control.....	816
35.4.17	Polarity control.....	819
35.4.18	Initialization.....	820
35.4.19	Features priority.....	820
35.4.20	Channel trigger output.....	821
35.4.21	Initialization trigger.....	822
35.4.22	Capture Test mode.....	824
35.4.23	DMA.....	825
35.4.24	Dual Edge Capture mode.....	826
35.4.25	Quadrature Decoder mode.....	833
35.4.26	BDM mode.....	838
35.4.27	Intermediate load.....	839
35.4.28	Global time base (GTB).....	841
35.5	Reset overview.....	843
35.6	FTM Interrupts.....	844
35.6.1	Timer Overflow Interrupt.....	845
35.6.2	Channel (n) Interrupt.....	845
35.6.3	Fault Interrupt.....	845
35.7	Initialization Procedure.....	845

Section number	Title	Page
<b>Chapter 36</b>		
<b>Periodic Interrupt Timer (PIT)</b>		
36.1	Introduction.....	847
36.1.1	Block diagram.....	847
36.1.2	Features.....	848
36.2	Signal description.....	848
36.3	Memory map/register description.....	849
36.3.1	PIT Module Control Register (PIT_MCR).....	849
36.3.2	Timer Load Value Register (PIT_LDVAL <sub>n</sub> ).....	851
36.3.3	Current Timer Value Register (PIT_CVAL <sub>n</sub> ).....	851
36.3.4	Timer Control Register (PIT_TCTRL <sub>n</sub> ).....	852
36.3.5	Timer Flag Register (PIT_TFLG <sub>n</sub> ).....	852
36.4	Functional description.....	853
36.4.1	General operation.....	853
36.4.2	Interrupts.....	855
36.4.3	Chained timers.....	855
36.5	Initialization and application information.....	855
36.6	Example configuration for chained timers.....	856
<b>Chapter 37</b>		
<b>Low-Power Timer (LPTMR)</b>		
37.1	Introduction.....	859
37.1.1	Features.....	859
37.1.2	Modes of operation.....	859
37.2	LPTMR signal descriptions.....	860
37.2.1	Detailed signal descriptions.....	860
37.3	Memory map and register definition.....	860
37.3.1	Low Power Timer Control Status Register (LPTMR <sub>x</sub> _CSR).....	861
37.3.2	Low Power Timer Prescale Register (LPTMR <sub>x</sub> _PSR).....	862
37.3.3	Low Power Timer Compare Register (LPTMR <sub>x</sub> _CMR).....	864

Section number	Title	Page
37.3.4	Low Power Timer Counter Register (LPTMR <sub>x</sub> _CNR).....	864
37.4	Functional description.....	865
37.4.1	LPTMR power and reset.....	865
37.4.2	LPTMR clocking.....	865
37.4.3	LPTMR prescaler/glitch filter.....	865
37.4.4	LPTMR compare.....	867
37.4.5	LPTMR counter.....	867
37.4.6	LPTMR hardware trigger.....	868
37.4.7	LPTMR interrupt.....	868

## Chapter 38 Serial Peripheral Interface (SPI)

38.1	Introduction.....	869
38.1.1	Block Diagram.....	869
38.1.2	Features.....	870
38.1.3	Interface configurations.....	872
38.1.4	Modes of Operation.....	872
38.2	Module signal descriptions.....	874
38.2.1	PCS0/SS—Peripheral Chip Select/Slave Select.....	874
38.2.2	PCS1–PCS3—Peripheral Chip Selects 1–3.....	875
38.2.3	PCS4—Peripheral Chip Select 4.....	875
38.2.4	PCS5/PCSS—Peripheral Chip Select 5/Peripheral Chip Select Strobe.....	875
38.2.5	SCK—Serial Clock.....	875
38.2.6	SIN—Serial Input.....	875
38.2.7	SOUT—Serial Output.....	876
38.3	Memory Map/Register Definition.....	876
38.3.1	Module Configuration Register (SPI <sub>x</sub> _MCR).....	878
38.3.2	Transfer Count Register (SPI <sub>x</sub> _TCR).....	881
38.3.3	Clock and Transfer Attributes Register (In Master Mode) (SPI <sub>x</sub> _CTAR <sub>n</sub> ).....	881
38.3.4	Clock and Transfer Attributes Register (In Slave Mode) (SPI <sub>x</sub> _CTAR <sub>n</sub> _SLAVE).....	886



Section number	Title	Page
38.3.5	Status Register (SPLx_SR).....	888
38.3.6	DMA/Interrupt Request Select and Enable Register (SPLx_RSER).....	891
38.3.7	PUSH TX FIFO Register In Master Mode (SPLx_PUSHR).....	893
38.3.8	PUSH TX FIFO Register In Slave Mode (SPLx_PUSHR_SLAVE).....	894
38.3.9	POP RX FIFO Register (SPLx_POPR).....	895
38.3.10	Transmit FIFO Registers (SPLx_TXFRn).....	896
38.3.11	Receive FIFO Registers (SPLx_RXFRn).....	896
38.4	Functional description.....	897
38.4.1	Start and Stop of module transfers.....	898
38.4.2	Serial Peripheral Interface (SPI) configuration.....	898
38.4.3	Module baud rate and clock delay generation.....	902
38.4.4	Transfer formats.....	906
38.4.5	Continuous Serial Communications Clock.....	911
38.4.6	Slave Mode Operation Constraints.....	912
38.4.7	Interrupts/DMA requests.....	913
38.4.8	Power saving features.....	915
38.5	Initialization/application information.....	916
38.5.1	How to manage queues.....	916
38.5.2	Switching Master and Slave mode.....	917
38.5.3	Initializing Module in Master/Slave Modes.....	917
38.5.4	Baud rate settings.....	918
38.5.5	Delay settings.....	918
38.5.6	Calculation of FIFO pointer addresses.....	919

## Chapter 39 Inter-Integrated Circuit (I2C)

39.1	Introduction.....	923
39.1.1	Features.....	923
39.1.2	Modes of operation.....	924
39.1.3	Block diagram.....	924

Section number	Title	Page
39.2	I2C signal descriptions.....	925
39.3	Memory map/register definition.....	926
39.3.1	I2C Address Register 1 (I2Cx_A1).....	926
39.3.2	I2C Frequency Divider register (I2Cx_F).....	927
39.3.3	I2C Control Register 1 (I2Cx_C1).....	928
39.3.4	I2C Status register (I2Cx_S).....	930
39.3.5	I2C Data I/O register (I2Cx_D).....	931
39.3.6	I2C Control Register 2 (I2Cx_C2).....	932
39.3.7	I2C Programmable Input Glitch Filter Register (I2Cx_FLT).....	933
39.3.8	I2C Range Address register (I2Cx_RA).....	934
39.3.9	I2C SMBus Control and Status register (I2Cx_SMB).....	935
39.3.10	I2C Address Register 2 (I2Cx_A2).....	937
39.3.11	I2C SCL Low Timeout Register High (I2Cx_SLTH).....	937
39.3.12	I2C SCL Low Timeout Register Low (I2Cx_SLTL).....	937
39.4	Functional description.....	938
39.4.1	I2C protocol.....	938
39.4.2	10-bit address.....	943
39.4.3	Address matching.....	945
39.4.4	System management bus specification.....	946
39.4.5	Resets.....	948
39.4.6	Interrupts.....	948
39.4.7	Programmable input glitch filter.....	951
39.4.8	Address matching wake-up.....	951
39.4.9	DMA support.....	952
39.5	Initialization/application information.....	953

## Chapter 40 Universal Asynchronous Receiver/Transmitter (UART)

40.1	Introduction.....	957
40.1.1	Features.....	957

Section number	Title	Page
40.1.2	Modes of operation.....	958
40.2	UART signal descriptions.....	959
40.2.1	Detailed signal descriptions.....	960
40.3	Memory map and registers.....	961
40.3.1	UART Baud Rate Registers: High (UARTx_BDH).....	963
40.3.2	UART Baud Rate Registers: Low (UARTx_BDL).....	964
40.3.3	UART Control Register 1 (UARTx_C1).....	964
40.3.4	UART Control Register 2 (UARTx_C2).....	966
40.3.5	UART Status Register 1 (UARTx_S1).....	967
40.3.6	UART Status Register 2 (UARTx_S2).....	970
40.3.7	UART Control Register 3 (UARTx_C3).....	972
40.3.8	UART Data Register (UARTx_D).....	973
40.3.9	UART Match Address Registers 1 (UARTx_MA1).....	975
40.3.10	UART Match Address Registers 2 (UARTx_MA2).....	975
40.3.11	UART Control Register 4 (UARTx_C4).....	975
40.3.12	UART Control Register 5 (UARTx_C5).....	976
40.3.13	UART Extended Data Register (UARTx_ED).....	977
40.3.14	UART Modem Register (UARTx_MODEM).....	978
40.3.15	UART Infrared Register (UARTx_IR).....	979
40.3.16	UART FIFO Parameters (UARTx_PFIFO).....	980
40.3.17	UART FIFO Control Register (UARTx_CFIFO).....	981
40.3.18	UART FIFO Status Register (UARTx_SFIFO).....	982
40.3.19	UART FIFO Transmit Watermark (UARTx_TWFIFO).....	983
40.3.20	UART FIFO Transmit Count (UARTx_TCFIFO).....	984
40.3.21	UART FIFO Receive Watermark (UARTx_RWFIFO).....	984
40.3.22	UART FIFO Receive Count (UARTx_RCFIFO).....	985

Section number	Title	Page
40.4	Functional description.....	985
40.4.1	Transmitter.....	985
40.4.2	Receiver.....	990
40.4.3	Baud rate generation.....	1002
40.4.4	Data format.....	1004
40.4.5	Single-wire operation.....	1008
40.4.6	Loop operation.....	1008
40.4.7	Infrared interface.....	1009
40.5	Reset.....	1010
40.6	System level interrupt sources.....	1010
40.6.1	RXEDGIF description.....	1010
40.7	DMA operation.....	1011
40.8	Application information.....	1012
40.8.1	Initialization sequence.....	1012
40.8.2	Overrun (OR) flag implications.....	1013
40.8.3	Match address registers.....	1014
40.8.4	Modem feature.....	1014
40.8.5	IrDA minimum pulse width.....	1015
40.8.6	Legacy and reverse compatibility considerations.....	1016

## Chapter 41 General-Purpose Input/Output (GPIO)

41.1	Introduction.....	1017
41.1.1	Features.....	1017
41.1.2	Modes of operation.....	1017
41.1.3	GPIO signal descriptions.....	1018
41.2	Memory map and register definition.....	1019
41.2.1	Port Data Output Register (GPIOx_PDOR).....	1021
41.2.2	Port Set Output Register (GPIOx_PSOR).....	1022
41.2.3	Port Clear Output Register (GPIOx_PCOR).....	1022

Section number	Title	Page
41.2.4	Port Toggle Output Register (GPIOx_PTOR).....	1023
41.2.5	Port Data Input Register (GPIOx_PDIR).....	1023
41.2.6	Port Data Direction Register (GPIOx_PDDR).....	1024
41.3	Functional description.....	1024
41.3.1	General-purpose input.....	1024
41.3.2	General-purpose output.....	1024

## Chapter 42 JTAG Controller (JTAGC)

42.1	Introduction.....	1027
42.1.1	Block diagram.....	1027
42.1.2	Features.....	1028
42.1.3	Modes of operation.....	1028
42.2	External signal description.....	1030
42.2.1	TCK—Test clock input.....	1030
42.2.2	TDI—Test data input.....	1030
42.2.3	TDO—Test data output.....	1030
42.2.4	TMS—Test mode select.....	1030
42.3	Register description.....	1031
42.3.1	Instruction register.....	1031
42.3.2	Bypass register.....	1031
42.3.3	Device identification register.....	1031
42.3.4	Boundary scan register.....	1032
42.4	Functional description.....	1033
42.4.1	JTAGC reset configuration.....	1033
42.4.2	IEEE 1149.1-2001 (JTAG) Test Access Port.....	1033
42.4.3	TAP controller state machine.....	1033
42.4.4	JTAGC block instructions.....	1035
42.4.5	Boundary scan.....	1038
42.5	Initialization/Application information.....	1038



# Chapter 1

## About This Document

### 1.1 Overview

#### 1.1.1 Purpose

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

#### 1.1.2 Audience

This document is intended 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 <i>0b</i> .
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 <i>0x</i> .

## 1.2.2 Typographic notation

The following typographic notation is used throughout this document:

Example	Description
<i>placeholder, x</i>	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: <ul style="list-style-type: none"> <li>• 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.</li> <li>• A continuous range of individual signals of a bus For example, XAD[7:0] refers to signals 7–0 of the XAD bus.</li> </ul>

## 1.2.3 Special terms

The following terms have special meanings:

Term	Meaning
asserted	Refers to the state of a signal as follows: <ul style="list-style-type: none"> <li>• An active-high signal is asserted when high (1).</li> <li>• An active-low signal is asserted when low (0).</li> </ul>
deasserted	Refers to the state of a signal as follows: <ul style="list-style-type: none"> <li>• An active-high signal is deasserted when low (0).</li> <li>• An active-low signal is deasserted when high (1).</li> </ul> <p>In some cases, deasserted signals are described as <i>negated</i>.</p>
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®-M4 core	<ul style="list-style-type: none"><li>• 32-bit MCU core from ARM's Cortex-M class adding DSP instructions and single-precision floating point unit based on ARMv7 architecture</li></ul>
System	<ul style="list-style-type: none"><li>• System integration module</li><li>• Power management and mode controllers<ul style="list-style-type: none"><li>• Multiple power modes available based on high speed run, run, wait, stop, and power-down modes</li></ul></li><li>• Low-leakage wakeup unit</li><li>• Miscellaneous control module</li><li>• Crossbar switch</li><li>• Peripheral bridge</li><li>• Direct memory access (DMA) controller with multiplexer to increase available DMA requests.</li><li>• External watchdog monitor</li><li>• Watchdog</li></ul>
Memories	<ul style="list-style-type: none"><li>• Internal memories include:<ul style="list-style-type: none"><li>• Program flash memory</li><li>• SRAM</li></ul></li></ul>
Clocks	<ul style="list-style-type: none"><li>• Multiple clock generation options available from internally- and externally-generated clocks</li><li>• System oscillator to provide clock source for the MCU</li></ul>
Security	<ul style="list-style-type: none"><li>• Cyclic Redundancy Check module for error detection</li></ul>

*Table continues on the next page...*

**Table 2-1. Module functional categories (continued)**

Module category	Description
Analog	<ul style="list-style-type: none"> <li>• High speed analog-to-digital converter</li> <li>• Comparator</li> <li>• Digital-to-analog converter</li> <li>• Internal voltage reference</li> <li>• Bandgap voltage reference</li> </ul>
Timers	<ul style="list-style-type: none"> <li>• Programmable delay block</li> <li>• FlexTimers</li> <li>• Periodic interrupt timer</li> <li>• Low power timer</li> </ul>
Communications	<ul style="list-style-type: none"> <li>• Serial peripheral interface</li> <li>• Inter-integrated circuit (I<sup>2</sup>C)</li> <li>• UART</li> </ul>
Human-Machine Interfaces (HMI)	<ul style="list-style-type: none"> <li>• General purpose input/output controller</li> </ul>

## 2.2.1 ARM® Cortex®-M4 Core Modules

The following core modules are available on this device.

**Table 2-2. Core modules**

Module	Description
ARM Cortex-M4	The ARM® Cortex®-M4 is the newest member of the Cortex M Series of processors targeting microcontroller cores focused on very cost sensitive, deterministic, interrupt driven environments. The Cortex M4 processor is based on the ARMv7 Architecture and Thumb®-2 ISA and is upward compatible with the Cortex M3, Cortex M1, and Cortex M0 architectures. Cortex M4 improvements include an ARMv7 Thumb-2 DSP (ported from the ARMv7-A/R profile architectures) providing 32-bit instructions with SIMD (single instruction multiple data) DSP style multiply-accumulates and saturating arithmetic.
Floating point unit (FPU)	A single-precision floating point unit (FPU) that is compliant to the <i>IEEE Standard for Floating-Point Arithmetic</i> (IEEE 754).
NVIC	<p>The ARMv7-M exception model and nested-vector interrupt controller (NVIC) implement a relocatable vector table supporting many external interrupts, a single non-maskable interrupt (NMI), and priority levels.</p> <p>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.</p>
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.

Table continues on the next page...

**Table 2-2. Core modules (continued)**

Module	Description
Debug interfaces	<p>Most of this device's debug is based on the ARM CoreSight™ architecture. Four debug interfaces are supported:</p> <ul style="list-style-type: none"> <li>• IEEE 1149.1 JTAG</li> <li>• IEEE 1149.7 JTAG (cJTAG)</li> <li>• Serial Wire Debug (SWD)</li> <li>• ARM Real-Time Trace Interface</li> </ul>

## 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 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
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.
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-bit, 16-bit, 32-bit, 16-byte and 32-byte 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.

## 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	<ul style="list-style-type: none"> <li>Program flash memory — non-volatile flash memory that can execute program code</li> </ul>
Flash memory controller	Manages the interface between the device and the on-chip flash memory.
SRAM	Internal system RAM. Partial SRAM kept powered in LLS2 and VLLS2 low leakage mode.

## 2.2.4 Clocks

The following clock modules are available on this device.

**Table 2-5. Clock modules**

Module	Description
Multi-clock generator (MCG)	<p>The MCG provides several clock sources for the MCU that include:</p> <ul style="list-style-type: none"> <li>Frequency-locked loop (FLL) — Digitally-controlled oscillator (DCO)</li> <li>Internal reference clocks — Can be used as a clock source for other on-chip peripherals</li> </ul>
48 MHz Internal Reference Clock (IRC48M)	The IRC48M provides an internally generated clock source.
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 modules are 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.
6-bit digital-to-analog converters (DAC)	64-tap resistor ladder network which provides a selectable voltage reference for applications where voltage reference is needed.
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.
Voltage reference (VREF)	Supplies an accurate voltage output that is trimmable in 0.5 mV steps. The VREF can be used in medical applications, such as glucose meters, to provide a reference voltage to biosensors or as a reference to analog peripherals, such as the ADC, DAC, or CMP.

## 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)	<ul style="list-style-type: none"> <li>• 16-bit resolution</li> <li>• 3-bit prescaler</li> <li>• Positive transition of trigger event signal initiates the counter</li> <li>• Supports two triggered delay output signals, each with an independently-controlled delay from the trigger event</li> <li>• 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.</li> <li>• Continuous-pulse output or single-shot mode supported, each output is independently enabled, with possible trigger events</li> <li>• Supports bypass mode</li> <li>• Supports DMA</li> </ul>

*Table continues on the next page...*

**Table 2-8. Timer modules (continued)**

Module	Description
Flexible timer modules (FTM)	<ul style="list-style-type: none"> <li>• Selectable FTM source clock, programmable prescaler</li> <li>• 16-bit counter supporting free-running or initial/final value, and counting is up or up-down</li> <li>• Input capture, output compare, and edge-aligned and center-aligned PWM modes</li> <li>• Operation of FTM channels as pairs with equal outputs, pairs with complimentary outputs, or independent channels with independent outputs</li> <li>• Deadtime insertion is available for each complementary pair</li> <li>• Generation of hardware triggers</li> <li>• Software control of PWM outputs</li> <li>• Up to 4 fault inputs for global fault control</li> <li>• Configurable channel polarity</li> <li>• Programmable interrupt on input capture, reference compare, overflowed counter, or detected fault condition</li> <li>• Quadrature decoder with input filters, relative position counting, and interrupt on position count or capture of position count on external event</li> <li>• DMA support for FTM events</li> </ul>
Periodic interrupt timers (PIT)	<ul style="list-style-type: none"> <li>• Four general purpose interrupt timers</li> <li>• Interrupt timers for triggering ADC conversions</li> <li>• 32-bit counter resolution</li> <li>• DMA support</li> </ul>
Low-power timer (LPTimer)	<ul style="list-style-type: none"> <li>• Selectable clock for prescaler/glitch filter of 1 kHz (internal LPO), 32.768 kHz (external crystal), or internal reference clock</li> <li>• Configurable Glitch Filter or Prescaler with 16-bit counter</li> <li>• 16-bit time or pulse counter with compare</li> <li>• Interrupt generated on Timer Compare</li> <li>• Hardware trigger generated on Timer Compare</li> </ul>

## 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

## 2.2.9 Human-machine interfaces

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

**Table 2-10. HMI modules**

Module	Description
<a href="#">General purpose input/output (GPIO)</a>	All general purpose input or output (GPIO) pins are capable of interrupt and DMA request generation.

## 2.3 Orderable part numbers

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

**Table 2-11. Orderable part numbers summary**

Freescale part number	CPU frequency	Pin count	Package	Program flash	SRAM	GPIO
MK02FN128VLH10	100 MHz	64	LQFP	128 KB	16 KB	46
MK02FN128VLF10 <sup>1</sup>	100 MHz	48	LQFP	128 KB	16 KB	35
MK02FN128VFM10	100 MHz	32	QFN	128 KB	16 KB	26
MK02FN64VLH10	100 MHz	64	LQFP	64 KB	16 KB	46
MK02FN64VLF10 <sup>1</sup>	100 MHz	48	LQFP	64 KB	16 KB	35
MK02FN64VFM10	100 MHz	32	QFN	64 KB	16 KB	26

1. This part number is subject to removal.





# Chapter 3

## Chip Configuration

### 3.1 Introduction

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

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

### 3.2 Core modules

#### 3.2.1 ARM Cortex-M4 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](http://arm.com).

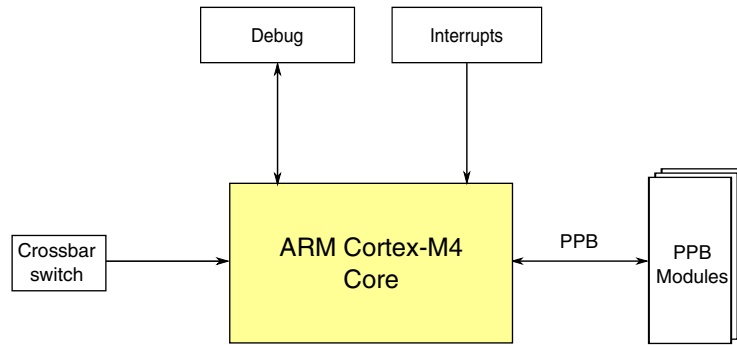


Figure 3-1. Core configuration

Table 3-1. Reference links to related information

Topic	Related module	Reference
Full description	ARM Cortex-M4 core	<a href="#">ARM Cortex-M4 Technical Reference Manual</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
System/instruction/data bus module	Crossbar switch	<a href="#">Crossbar switch</a>
Debug	IEEE 1149.1 JTAG IEEE 1149.7 JTAG (cJTAG) Serial Wire Debug (SWD) ARM Real-Time Trace Interface	<a href="#">Debug</a>
Interrupts	Nested Vectored Interrupt Controller (NVIC)	<a href="#">NVIC</a>
Private Peripheral Bus (PPB) module	Miscellaneous Control Module (MCM)	<a href="#">MCM</a>
Private Peripheral Bus (PPB) module	Single-precision floating point unit (FPU)	<a href="#">FPU</a>

### 3.2.1.1 Buses, interconnects, and interfaces

The ARM Cortex-M4 core has four buses as described in the following table.

Bus name	Description
Instruction code (ICODE) bus	The ICODE and DCODE buses are muxed. This muxed bus is called the CODE bus and is connected to the crossbar switch via a single master port.
Data code (DCODE) bus	
System bus	The system bus is connected to a separate master port on the crossbar.

Table continues on the next page...

Bus name	Description
Private peripheral (PPB) bus	The PPB provides access to these modules: <ul style="list-style-type: none"> <li>• ARM modules such as the NVIC, ITM, DWT, FBP, and ROM table</li> <li>• Freescale Miscellaneous Control Module (MCM)</li> </ul>

### 3.2.1.2 System Tick Timer

The System Tick Timer's clock source is always the core clock, FCLK. This results in the following:

- The CLKSOURCE bit in SysTick Control and Status register is always set to select the core clock.
- Because the timing reference (FCLK) is a variable frequency, the TENMS bit in the SysTick Calibration Value Register is always zero.
- The NOREF bit in SysTick Calibration Value Register is always set, implying that FCLK is the only available source of reference timing.

### 3.2.1.3 Debug facilities

This device has extensive debug capabilities including run control and tracing capabilities. The standard ARM debug port that supports JTAG and SWD interfaces. Also the cJTAG interface is supported on this device.

### 3.2.1.4 Core privilege 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

## 3.2.2 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](http://arm.com).

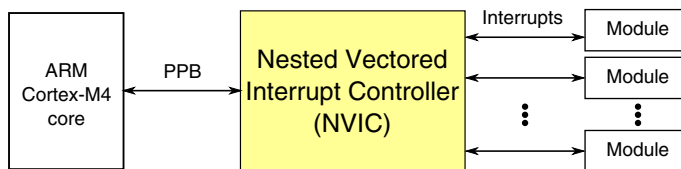


Figure 3-2. NVIC configuration

Table 3-2. Reference links to related information

Topic	Related module	Reference
Full description	Nested Vectored Interrupt Controller (NVIC)	<a href="#">ARM Cortex-M4 Technical Reference Manual</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Private Peripheral Bus (PPB)	ARM Cortex-M4 core	<a href="#">ARM Cortex-M4 core</a>

### 3.2.2.1 Interrupt priority levels

This device supports 16 priority levels for interrupts. Therefore, in the NVIC each source in the IPR registers contains 4 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				IRQ2				0				IRQ1				0				IRQ0				0			
W																																

### 3.2.2.2 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.

### 3.2.2.3 Interrupt channel assignments

The interrupt source 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 3-4. Interrupt vector assignments**

Address	Vector	IRQ <sup>1</sup>	NVIC non-IPR register number <sup>2</sup>	NVIC IPR register number <sup>3</sup>	Source module	Source description
<b>ARM Core System Handler Vectors</b>						
0x0000_0000	0	—	—	—	ARM core	Initial Stack Pointer
0x0000_0004	1	—	—	—	ARM core	Initial Program Counter
0x0000_0008	2	—	—	—	ARM core	Non-maskable Interrupt (NMI)
0x0000_000C	3	—	—	—	ARM core	Hard Fault
0x0000_0010	4	—	—	—	ARM core	MemManage Fault
0x0000_0014	5	—	—	—	ARM core	Bus Fault
0x0000_0018	6	—	—	—	ARM core	Usage Fault
0x0000_001C	7	—	—	—	—	—
0x0000_0020	8	—	—	—	—	—
0x0000_0024	9	—	—	—	—	—
0x0000_0028	10	—	—	—	—	—
0x0000_002C	11	—	—	—	ARM core	Supervisor call (SVCall)
0x0000_0030	12	—	—	—	ARM core	Debug Monitor
0x0000_0034	13	—	—	—	—	—
0x0000_0038	14	—	—	—	ARM core	Pendable request for system service (PendableSrvReq)
0x0000_003C	15	—	—	—	ARM core	System tick timer (SysTick)
<b>Non-Core Vectors</b>						
0x0000_0040	16	0	0	0	DMA	DMA channel 0 transfer complete
0x0000_0044	17	1	0	0	DMA	DMA channel 1 transfer complete
0x0000_0048	18	2	0	0	DMA	DMA channel 2 transfer complete
0x0000_004C	19	3	0	0	DMA	DMA channel 3 transfer complete
0x0000_0050	20	4	0	1	—	—
0x0000_0054	21	5	0	1	—	—
0x0000_0058	22	6	0	1	—	—
0x0000_005C	23	7	0	1	—	—
0x0000_0060	24	8	0	2	—	—
0x0000_0064	25	9	0	2	—	—
0x0000_0068	26	10	0	2	—	—
0x0000_006C	27	11	0	2	—	—

Table continues on the next page...

Table 3-4. Interrupt vector assignments (continued)

Address	Vector	IRQ <sup>1</sup>	NVIC non-IPR register number <sup>2</sup>	NVIC IPR register number <sup>3</sup>	Source module	Source description
0x0000_0070	28	12	0	3	—	—
0x0000_0074	29	13	0	3	—	—
0x0000_0078	30	14	0	3	—	—
0x0000_007C	31	15	0	3	—	—
0x0000_0080	32	16	0	4	DMA	DMA error interrupt channels 0-3
0x0000_0084	33	17	0	4	MCM	FPU sources
0x0000_0088	34	18	0	4	Flash memory	Command complete
0x0000_008C	35	19	0	4	Flash memory	Read collision
0x0000_0090	36	20	0	5	Mode Controller	Low-voltage detect, low-voltage warning
0x0000_0094	37	21	0	5	LLWU	Low Leakage Wakeup  <b>NOTE:</b> The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit stop mode on an LLS recovery.
0x0000_0098	38	22	0	5	WDOG or EWM	Both watchdog modules share this interrupt.
0x0000_009C	39	23	0	5	—	—
0x0000_00A0	40	24	0	6	I <sup>2</sup> C0	—
0x0000_00A4	41	25	0	6	—	—
0x0000_00A8	42	26	0	6	SPI0	Single interrupt vector for all sources
0x0000_00AC	43	27	0	6	—	—
0x0000_00B0	44	28	0	7	—	—
0x0000_00B4	45	29	0	7	—	—
0x0000_00B8	46	30	0	7	—	—
0x0000_00BC	47	31	0	7	UART0	Single interrupt vector for UART status sources
0x0000_00C0	48	32	1	8	UART0	Single interrupt vector for UART error sources
0x0000_00C4	49	33	1	8	UART1	Single interrupt vector for UART status sources
0x0000_00C8	50	34	1	8	UART1	Single interrupt vector for UART error sources
0x0000_00CC	51	35	1	8	—	—
0x0000_00D0	52	36	1	9	—	—
0x0000_00D4	53	37	1	9	—	—
0x0000_00D8	54	38	1	9	—	—
0x0000_00DC	55	39	1	9	ADC0	—

Table continues on the next page...

Table 3-4. Interrupt vector assignments (continued)

Address	Vector	IRQ <sup>1</sup>	NVIC non-IPR register number <sup>2</sup>	NVIC IPR register number <sup>3</sup>	Source module	Source description
0x0000_00E0	56	40	1	10	CMP0	—
0x0000_00E4	57	41	1	10	CMP1	—
0x0000_00E8	58	42	1	10	FTM0	Single interrupt vector for all sources
0x0000_00EC	59	43	1	10	FTM1	Single interrupt vector for all sources
0x0000_00F0	60	44	1	11	FTM2	Single interrupt vector for all sources
0x0000_00F4	61	45	1	11	—	—
0x0000_00F8	62	46	1	11	Reserved	—
0x0000_00FC	63	47	1	11	Reserved	—
0x0000_0100	64	48	1	12	PIT	Channel 0
0x0000_0104	65	49	1	12	PIT	Channel 1
0x0000_0108	66	50	1	12	PIT	Channel 2
0x0000_010C	67	51	1	12	PIT	Channel 3
0x0000_0110	68	52	1	13	PDB	—
0x0000_0114	69	53	1	13	—	—
0x0000_0118	70	54	1	13	—	—
0x0000_011C	71	55	1	13	—	—
0x0000_0120	72	56	1	14	DAC0	—
0x0000_0124	73	57	1	14	MCG	—
0x0000_0128	74	58	1	14	Low Power Timer	—
0x0000_012C	75	59	1	14	Port control module	Pin detect (Port A)
0x0000_0130	76	60	1	15	Port control module	Pin detect (Port B)
0x0000_0134	77	61	1	15	Port control module	Pin detect (Port C)
0x0000_0138	78	62	1	15	Port control module	Pin detect (Port D)
0x0000_013C	79	63	1	15	Port control module	Pin detect (Port E)
0x0000_0140	80	64	2	16	Software	Software interrupt <sup>4</sup>
0x0000_0144	81	65	2	16	—	—
0x0000_0148	82	66	2	16	—	—
0x0000_014C	83	67	2	16	—	—
0x0000_0150	84	68	2	17	—	—
0x0000_0154	85	69	2	17	—	—
0x0000_0158	86	70	2	17	—	—
0x0000_015C	87	71	2	17	—	—
0x0000_0160	88	72	2	18	—	—
0x0000_0164	89	73	2	18	—	—

1. Indicates the NVIC's interrupt source number.

2. Indicates the NVIC's ISER, ICER, ISPR, ICPR, and IABR register number used for this IRQ. The equation to calculate this value is:  $IRQ \div 32$

3. Indicates the NVIC's IPR register number used for this IRQ. The equation to calculate this value is:  $IRQ \div 4$

4. This interrupt can only be pended or cleared via the NVIC registers.

### 3.2.2.3.1 Determining the bitfield and register location for configuring a particular interrupt

Suppose you need to configure the low-power timer (LPTMR) interrupt. The following table is an excerpt of the LPTMR row from [Interrupt channel assignments](#).

**Table 3-5. LPTMR interrupt vector assignment**

Address	Vector	IRQ <sup>1</sup>	NVIC non-IPR register number <sup>2</sup>	NVIC IPR register number <sup>3</sup>	Source module	Source description
0x0000_0128	74	58	1	14	Low Power Timer	—

1. Indicates the NVIC's interrupt source number.

2. Indicates the NVIC's ISER, ICER, ISPR, ICPR, and IABR register number used for this IRQ. The equation to calculate this value is:  $IRQ \div 32$

3. 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:
  - NVICISER1
  - NVICICER1
  - NVICISPR1
  - NVICICPR1
  - NVICIABR1
  - NVICIPR14
- To determine the particular IRQ's bitfield location within these particular registers:
  - NVICISER1, NVICICER1, NVICISPR1, NVICICPR1, NVICIABR1 bit location =  $IRQ \bmod 32 = 26$
  - NVICIPR14 bitfield starting location =  $8 * (IRQ \bmod 4) + 4 = 20$

Since the NVICIPR bitfields are 4-bit wide ([16 priority levels](#)), the NVICIPR14 bitfield range is 20-23

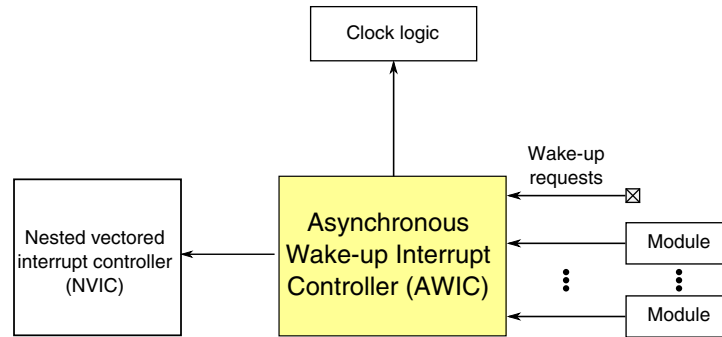
Therefore, the following bitfield locations are used to configure the LPTMR interrupts:

- NVICISER1[26]
- NVICICER1[26]
- NVICISPR1[26]
- NVICICPR1[26]
- NVICIABR1[26]
- NVICIPR14[23:20]



### 3.2.3 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](http://arm.com).



**Figure 3-3. Asynchronous Wake-up Interrupt Controller configuration**

**Table 3-6. Reference links to related information**

Topic	Related module	Reference
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
	Nested Vectored Interrupt Controller (NVIC)	<a href="#">NVIC</a>
Wake-up requests		<a href="#">AWIC wake-up sources</a>

#### 3.2.3.1 Wake-up sources

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

**Table 3-7. AWIC Partial Stop, Stop and VLPS Wake-up Sources**

Wake-up source	Description
Available system resets	RESET pin and WDOG when LPO is its clock source, and JTAG
Low-voltage detect	Power Mode Controller
Low-voltage warning	Power Mode Controller
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

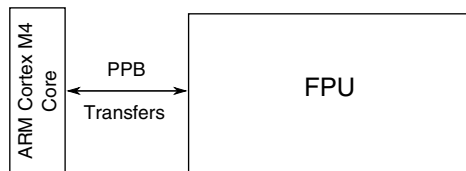
*Table continues on the next page...*

**Table 3-7. AWIC Partial Stop, Stop and VLPS Wake-up Sources (continued)**

Wake-up source	Description
CMPx	Since no system clocks are available, functionality is limited, trigger mode provides wakeup functionality with periodic sampling
I <sup>2</sup> C	Address match wakeup
UART	Active edge on RXD
LPTMR	Functional when using clock source which is active in Stop and VLPS modes
NMI	Non-maskable interrupt

### 3.2.4 FPU 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. FPU configuration**

**Table 3-8. Reference links to related information**

Topic	Related module	Reference
Full description	FPU	<a href="#">ARM Cortex-M4 Technical Reference Manual</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Power Management		<a href="#">Power Management</a>
Transfers Private Peripheral Bus (PPB)	ARM Cortex M4 core	<a href="#">ARM Cortex-M4 core</a>

### 3.2.5 JTAG 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.

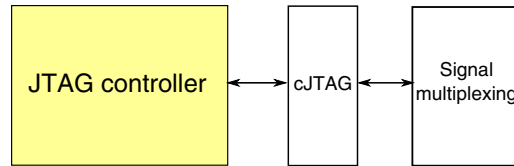


Figure 3-5. JTAGC Controller configuration

Table 3-9. Reference links to related information

Topic	Related module	Reference
Full description	JTAGC	<a href="#">JTAGC</a>
Signal multiplexing	Port control	<a href="#">Signal multiplexing</a>

## 3.3 System modules

### 3.3.1 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.

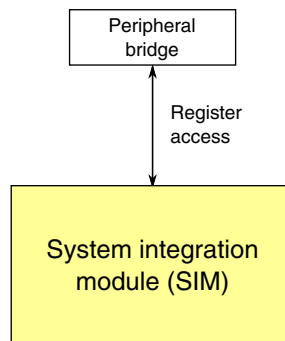


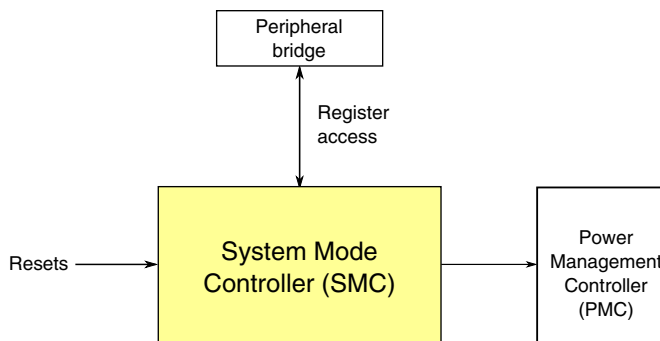
Figure 3-6. SIM configuration

Table 3-10. Reference links to related information

Topic	Related module	Reference
Full description	SIM	
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>

### 3.3.2 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 3-7. System Mode Controller configuration**

**Table 3-11. Reference links to related information**

Topic	Related module	Reference
Full description	System Mode Controller (SMC)	<a href="#">SMC</a>
System memory map		<a href="#">System memory map</a>
Power management		<a href="#">Power management</a>
	Power management controller (PMC)	<a href="#">PMC</a>
	Low-Leakage Wakeup Unit (LLWU)	<a href="#">LLWU</a>
	Reset Control Module (RCM)	<a href="#">Reset</a>

### 3.3.3 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.

**Table 3-12. Reference links to related information**

Topic	Related module	Reference
Full description	PMC	
System memory map		<a href="#">System memory map</a>
Power management		<a href="#">Power management</a>

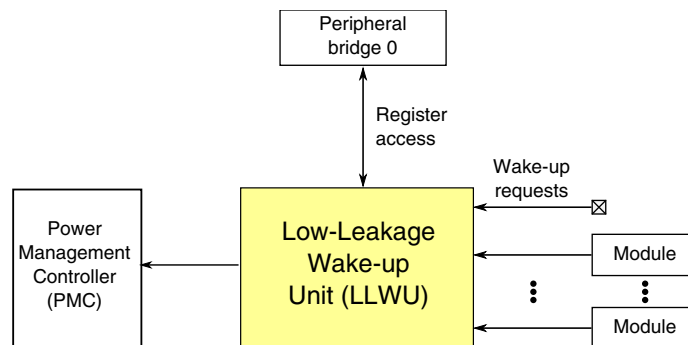
*Table continues on the next page...*

**Table 3-12. Reference links to related information (continued)**

Topic	Related module	Reference
Full description		
	Low-Leakage Wakeup Unit (LLWU)	<a href="#">LLWU</a>

### 3.3.4 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 3-8. Low-Leakage Wake-up Unit configuration****Table 3-13. Reference links to related information**

Topic	Related module	Reference
Full description	LLWU	
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management chapter</a>
	Power Management Controller (PMC)	<a href="#">Power Management Controller (PMC)</a>
	Mode Controller	
Wake-up requests		<a href="#">LLWU wake-up sources</a>

#### 3.3.4.1 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.

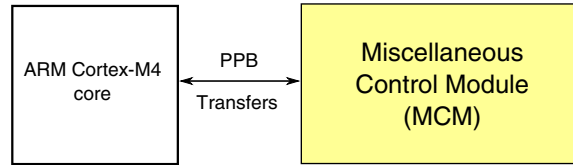
**Table 3-14. Wakeup sources for LLWU inputs**

Input	Wakeup source
LLWU_P0	PTE1/LLWU_P0 pin
LLWU_P1	Reserved
LLWU_P2	Reserved
LLWU_P3	PTA4/LLWU_P3 pin <sup>1</sup>
LLWU_P4	PTA13/LLWU_P4 pin
LLWU_P5	PTB0/LLWU_P5 pin
LLWU_P6	PTC1/LLWU_P6 pin
LLWU_P7	PTC3/LLWU_P7 pin
LLWU_P8	PTC4/LLWU_P8 pin
LLWU_P9	PTC5/LLWU_P9 pin
LLWU_P10	PTC6/LLWU_P10 pin
LLWU_P11	PTC11/LLWU_P11 pin
LLWU_P12	PTD0/LLWU_P12 pin
LLWU_P13	PTD2/LLWU_P13 pin
LLWU_P14	PTD4/LLWU_P14 pin
LLWU_P15	PTD6/LLWU_P15 pin
LLWU_M0IF	LPTMR <sup>2</sup>
LLWU_M1IF	CMP0 <sup>2</sup>
LLWU_M2IF	CMP1 <sup>2</sup>
LLWU_M3IF	Reserved
LLWU_M4IF	Reserved
LLWU_M5IF	Reserved
LLWU_M6IF	Reserved
LLWU_M7IF	Reserved

1. If NMI was enabled on entry to LLS/VLLS, asserting the NMI pin generates an NMI interrupt on exit from the low power mode. NMI can also be disabled via the FOPT[NMI\_DIS] bit.
2. Requires the peripheral and the peripheral interrupt to be enabled. The LLWU's WUME bit enables the internal module flag as a wakeup input. After wakeup, the flags are cleared based on the peripheral clearing mechanism.

### 3.3.5 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 3-15. Reference links to related information**

Topic	Related module	Reference
Full description	Miscellaneous control module (MCM)	<a href="#">MCM</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Transfers Private Peripheral Bus (PPB)	ARM Cortex-M4 core	<a href="#">ARM Cortex-M4 core</a>

### 3.3.6 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.

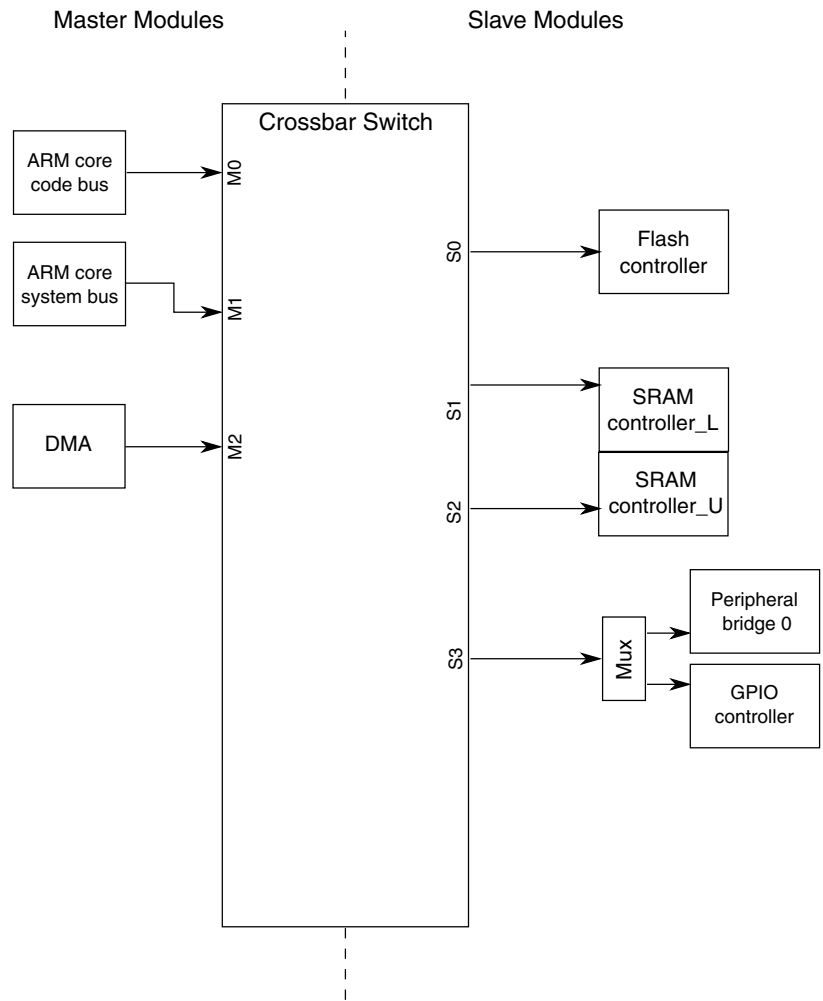


Figure 3-10. Crossbar-Light switch integration

Table 3-16. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	<a href="#">Crossbar Switch</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Crossbar switch master	ARM Cortex-M4 core	<a href="#">ARM Cortex-M4 core</a>
Crossbar switch master	DMA controller	<a href="#">DMA controller</a>
Crossbar switch slave	Flash	<a href="#">Flash</a>
Crossbar switch slave	Peripheral bridges	<a href="#">Peripheral bridge</a>
Crossbar switch slave	GPIO controller	<a href="#">GPIO controller</a>

### 3.3.6.1 Crossbar-Light Switch Master Assignments

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



Master module	Master port number
ARM core code bus	0
ARM core system bus	1
DMA	2

### 3.3.6.2 Crossbar-Light Switch Slave Assignments

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

Slave module	Slave port number
Flash memory controller	0
SRAM controllers	1,2
Peripheral bridge 0/GPIO <sup>1</sup>	3

1. See [System memory map](#) for access restrictions.

### 3.3.7 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.

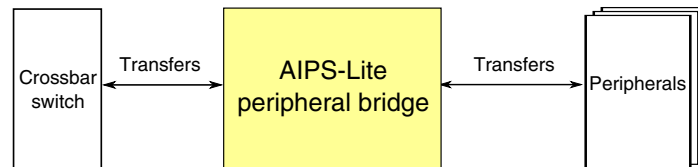


Figure 3-11. Peripheral bridge configuration

Table 3-17. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge (AIPS-Lite)	<a href="#">Peripheral bridge (AIPS-Lite)</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Crossbar switch	Crossbar switch	<a href="#">Crossbar switch</a>

#### 3.3.7.1 Number of peripheral bridges

This device contains one peripheral bridge.

### 3.3.7.2 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.

### 3.3.8 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.

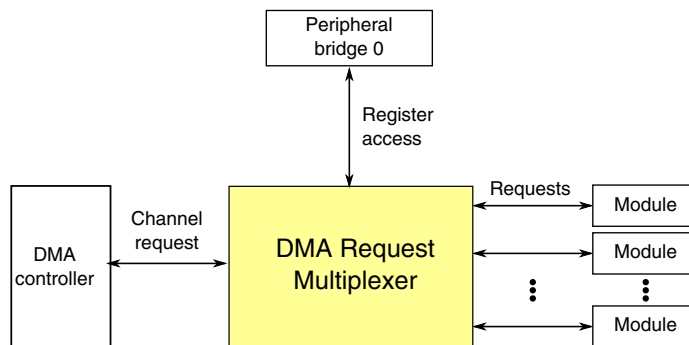


Figure 3-12. DMA request multiplexer configuration

Table 3-18. Reference links to related information

Topic	Related module	Reference
Full description	DMA request multiplexer	<a href="#">DMA Mux</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Channel request	DMA controller	<a href="#">DMA Controller</a>
Requests		<a href="#">DMA request sources</a>

#### 3.3.8.1 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 4 DMA channels. Because of the mux there is not a 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 3-19. DMA request sources - MUX 0**

Source number	Source module	Source description	Async DMA capable
0	—	Channel disabled <sup>1</sup>	
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	—	—	
13	—	—	
14	SPI0	Receive	
15	SPI0	Transmit	
16	Reserved	—	
17	Reserved	—	
18	I <sup>2</sup> C0	—	
19	Reserved	—	
20	FTM0	Channel 0	
21	FTM0	Channel 1	
22	FTM0	Channel 2	
23	FTM0	Channel 3	
24	FTM0	Channel 4	
25	FTM0	Channel 5	
26	Reserved	—	
27	Reserved	—	
28	FTM1	Channel 0	
29	FTM1	Channel 1	
30	FTM2	Channel 0	
31	FTM2	Channel 1	
32	Reserved	—	
33	Reserved	—	
34	Reserved	—	
35	Reserved	—	
36	Reserved	—	
37	Reserved	—	
38	Reserved	—	

*Table continues on the next page...*

**Table 3-19. DMA request sources - MUX 0 (continued)**

Source number	Source module	Source description	Async DMA capable
39	Reserved	—	
40	ADC0	—	Yes
41	Reserved	—	
42	CMP0	—	Yes
43	CMP1	—	Yes
44	Reserved	—	
45	DAC0	—	
46	Reserved	—	
47	Reserved	—	
48	PDB	—	
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	Reserved	—	
55	Reserved	—	
56	Reserved	—	
57	Reserved	—	
58	Reserved	—	
59	Reserved	—	
60	DMA MUX	Always enabled	
61	DMA MUX	Always enabled	
62	DMA MUX	Always enabled	
63	DMA MUX	Always enabled	

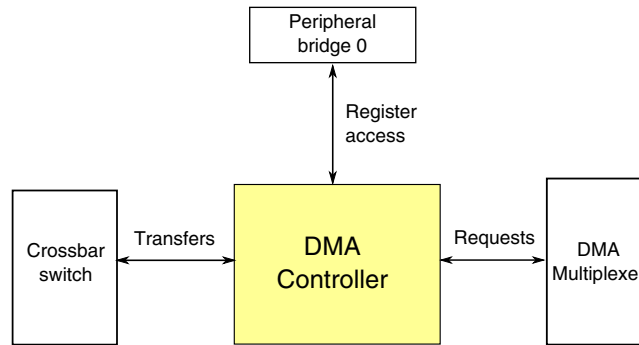
1. Configuring a DMA channel to select source 0 or any of the reserved sources disables that DMA channel.

### 3.3.8.2 DMA transfers via PIT trigger

The PIT module can trigger a DMA transfer on the first four DMA channels. The assignments are detailed at [PIT/DMA Periodic Trigger Assignments](#) .

### 3.3.9 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.



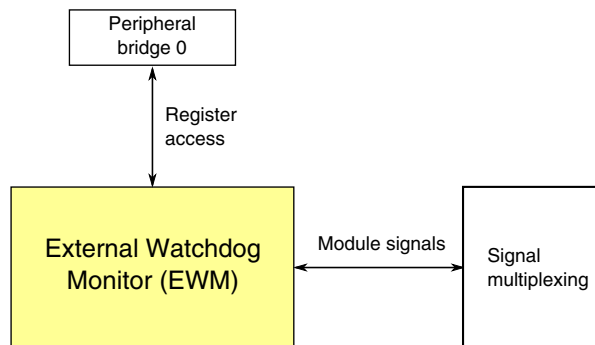
**Figure 3-13. DMA Controller configuration**

**Table 3-20. Reference links to related information**

Topic	Related module	Reference
Full description	DMA Controller	<a href="#">DMA Controller</a>
System memory map		<a href="#">System memory map</a>
Register access	Peripheral bridge (AIPS-Lite 0)	<a href="#">AIPS-Lite 0</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Transfers	Crossbar switch	<a href="#">Crossbar switch</a>

### 3.3.10 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 3-21. Reference links to related information**

Topic	Related module	Reference
Full description	External Watchdog Monitor (EWM)	<a href="#">EWM</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Signal multiplexing	Port Control Module	<a href="#">Signal multiplexing</a>

### 3.3.10.1 EWM clocks

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

**Table 3-22. EWM clock connections**

Module clock	Chip clock
Low Power Clock	1 kHz LPO Clock

### 3.3.10.2 EWM low-power modes

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

**Table 3-23. EWM low-power modes**

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

### 3.3.10.3 $\overline{\text{EWM\_OUT}}$ 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.

### 3.3.11 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.

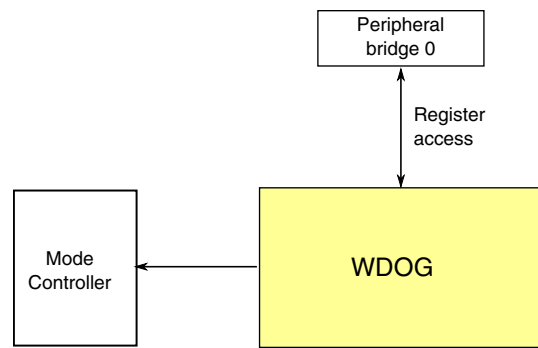


Figure 3-15. Watchdog configuration

Table 3-24. Reference links to related information

Topic	Related module	Reference
Full description	Watchdog	<a href="#">Watchdog</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
	Mode Controller (MC)	

#### 3.3.11.1 WDOG clocks

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

Table 3-25. WDOG clock connections

Module clock	Chip clock
LPO Oscillator	1 kHz LPO Clock

Table continues on the next page...

**Table 3-25. WDOG clock connections (continued)**

Module clock	Chip clock
Alt Clock	Bus Clock
Fast Test Clock	Bus Clock
System Bus Clock	Bus Clock

### 3.3.11.2 WDOG low-power modes

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

**Table 3-26. WDOG low-power modes**

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

## 3.4 Clock modules

### 3.4.1 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.



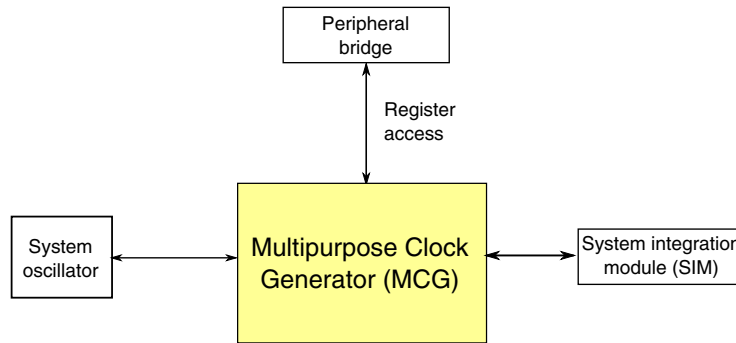


Figure 3-16. MCG configuration

Table 3-27. Reference links to related information

Topic	Related module	Reference
Full description	MCG	<a href="#">MCG</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Signal multiplexing	Port control	<a href="#">Signal multiplexing</a>

### 3.4.1.1 MCG oscillator clock input options

The MCG has multiple oscillator input clock sources. Within the context of the MCG these are all referred to as the external reference clock and selection is determined by MCG\_C7[OSCSEL] bitfield. The following table shows the chip-specific clock assignments for this bitfield.

Table 3-28. MCG Oscillator Reference Options

MCG_C7[OSCSEL]	MCG defined selection	Chip clock
00	OSCCLK0 - System Oscillator	OSCCLK - Undivided system oscillator output. Derived from external crystal circuit or directly from EXTAL.
01	OSC2/RTC Oscillator	Reserved
10	OSCCLK1 - Oscillator	IRC48MCLK. Derived from internal 48 MHz oscillator.
11	Reserved	—

See [Clock Distribution](#) for more details on these clocks.

#### NOTE

The MCG chapter has many references to the RTC oscillator source. On this device that clock source is not available and

register MCG\_C8 and other bit fields associated with this clock source should be treated as reserved.

### 3.4.2 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.

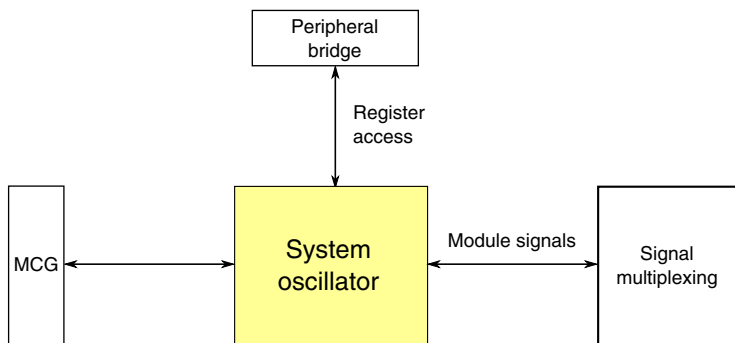


Figure 3-17. OSC configuration

Table 3-29. Reference links to related information

Topic	Related module	Reference
Full description	OSC	<a href="#">OSC</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Signal multiplexing	Port control	<a href="#">Signal multiplexing</a>
Full description	MCG	<a href="#">MCG</a>

#### 3.4.2.1 OSC modes of operation with MCG

The MCG's C2 register bits configure the oscillator frequency range. See the OSC and MCG chapters for more details.

## 3.5 Memories and memory interfaces

### 3.5.1 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.

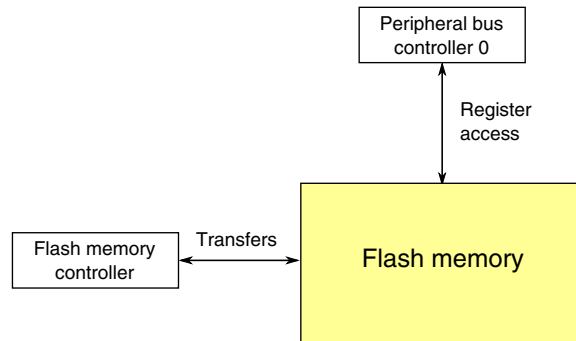


Figure 3-18. Flash memory configuration

Table 3-30. Reference links to related information

Topic	Related module	Reference
Full description	Flash memory	
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Transfers	Flash memory controller	<a href="#">Flash memory controller</a>
Register access	Peripheral bridge	<a href="#">Peripheral bridge</a>

#### 3.5.1.1 Flash memory types

This device contains the following types of flash memory:

- Program flash memory — non-volatile flash memory that can execute program code

#### 3.5.1.2 Flash Memory Sizes

The devices covered in this document contain:

- 1 block of program flash consisting of 2 KB sectors

The amounts of flash memory for the devices covered in this document are:

Device	Program flash (KB)	Block 0 address range
MK02FN128VLH10	128	0x0000_0000–0x0001_FFFF

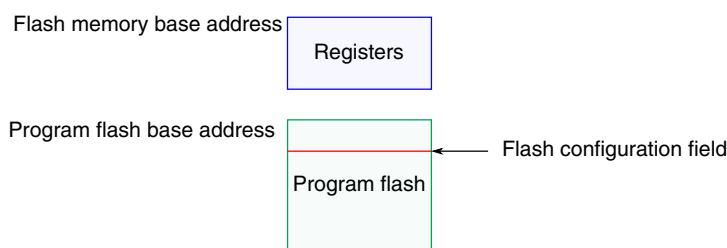
Table continues on the next page...

Device	Program flash (KB)	Block 0 address range
MK02FN128VLF10 <sup>1</sup>	128	0x0000_0000–0x0001_FFFF
MK02FN128VFM10	128	0x0000_0000–0x0001_FFFF
MK02FN64VLH10	64	0x0000_0000–0x0000_FFFF
MK02FN64VLF10 <sup>1</sup>	64	0x0000_0000–0x0000_FFFF
MK02FN64VFM10	64	0x0000_0000–0x0000_FFFF

1. This part number is subject to removal.

### 3.5.1.3 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 3-19. 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.

### 3.5.1.4 Flash Security

How flash security is implemented on this device is described in [Chip Security](#).

### 3.5.1.5 Flash Program Restrictions

The flash memory on this device should not be programmed or erased while operating in High Speed Run or VLPR power modes.

### 3.5.1.6 Flash Modes

The flash memory is always configured in NVM normal. There are no operating conditions in which the flash is configured for NVM special mode.

### 3.5.1.7 Erase All Flash Contents

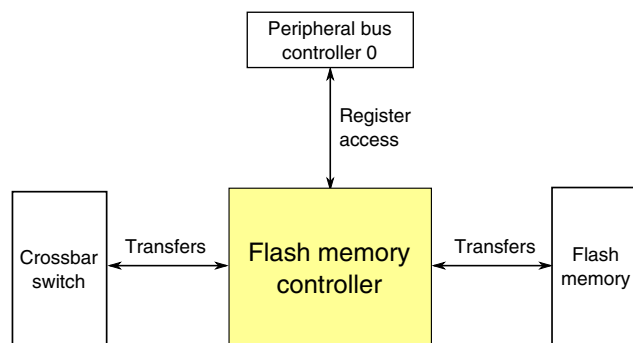
An Erase All Flash Blocks operation can be launched by software through a series of peripheral bus writes to flash registers. In addition the entire flash memory may be erased external to the flash memory from the SWJ-DP debug port by setting DAP\_CONTROL[0]. DAP\_STATUS[0] is set to indicate the mass erase command has been accepted. DAP\_STATUS[0] is cleared when the mass erase completes.

### 3.5.1.8 FTF\_FOPT Register

The flash memory's FTF\_FOPT register allows the user to customize the operation of the MCU at boot time. See [FOPT boot options](#) for details of its definition.

## 3.5.2 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.



**Figure 3-20. Flash memory controller configuration**

**Table 3-31. Reference links to related information**

Topic	Related module	Reference
Full description	Flash memory controller	<a href="#">Flash memory controller</a>

*Table continues on the next page...*

**Table 3-31. Reference links to related information (continued)**

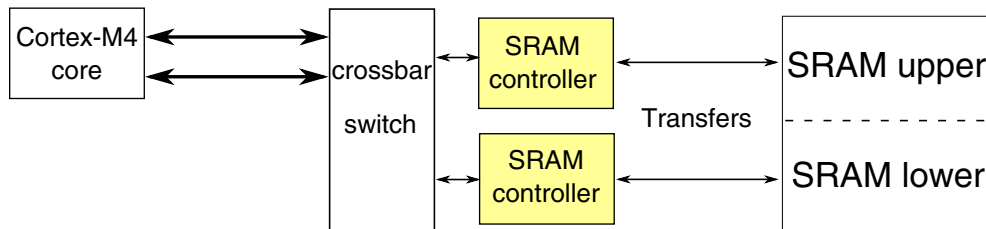
Topic	Related module	Reference
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Transfers	Flash memory	<a href="#">Flash memory</a>
Transfers	Crossbar switch	<a href="#">Crossbar Switch</a>
Register access	Peripheral bridge	<a href="#">Peripheral bridge</a>

### 3.5.2.1 Number of masters

The Flash Memory Controller supports up to eight crossbar switch masters. However, this device has a different number of crossbar switch masters. See [Crossbar-Light Switch Configuration](#) for details on the master port assignments.

### 3.5.3 SRAM Configuration

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



**Figure 3-21. SRAM configuration**

**Table 3-32. Reference links to related information**

Topic	Related module	Reference
Full description	SRAM	<a href="#">SRAM</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Transfers	SRAM controller	<a href="#">SRAM controller</a>
	ARM Cortex-M4 core	<a href="#">ARM Cortex-M4 core</a>

### 3.5.3.1 SRAM sizes

This device contains SRAM accessed by bus masters through the cross-bar switch. The on-chip SRAM is split into SRAM\_L and SRAM\_U regions where the SRAM\_L and SRAM\_U ranges form a contiguous block in the memory map anchored at address 0x2000\_0000. 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.

#### NOTE

Misaligned accesses across the 0x2000\_0000 boundary are not supported in the ARM Cortex-M4 architecture.

The amount of SRAM for the devices covered in this document is shown in the following table.

Device	SRAM_L size (KB)	SRAM_U size (KB)	Total SRAM (KB)	Address Range
MK02FN128VLH10	8	8	16	0x1FFF_E000-0x2000_1FFF
MK02FN128VLF10 <sup>1</sup>	8	8	16	0x1FFF_E000-0x2000_1FFF
MK02FN128VFM10	8	8	16	0x1FFF_E000-0x2000_1FFF
MK02FN64VLH10	8	8	16	0x1FFF_E000-0x2000_1FFF
MK02FN64VLF10 <sup>1</sup>	8	8	16	0x1FFF_E000-0x2000_1FFF
MK02FN64VFM10	8	8	16	0x1FFF_E000-0x2000_1FFF

1. This part number is subject to removal.

### 3.5.3.2 SRAM retention in low power modes

The SRAM is retained down to LLS3 and VLLS3 mode.

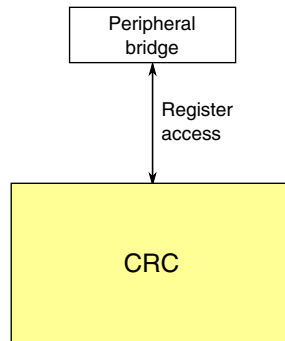
In LLS2 and VLLS2 the entire region of SRAM\_U from 0x2000\_0000 is powered.

In VLLS1 and VLLS0 no SRAM is retained.

## 3.6 Security

### 3.6.1 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 3-22. CRC configuration**

**Table 3-33. Reference links to related information**

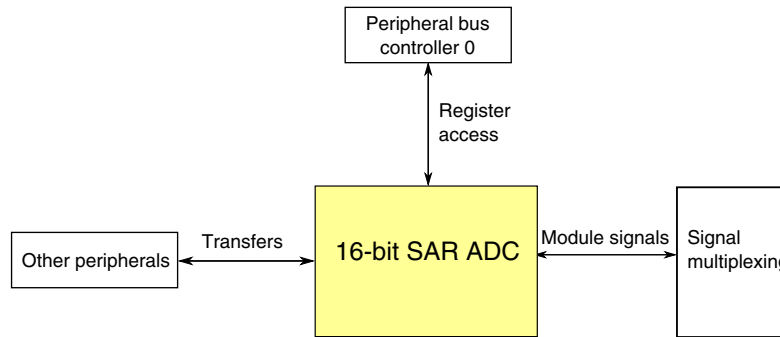
Topic	Related module	Reference
Full description	CRC	<a href="#">CRC</a>
System memory map		<a href="#">System memory map</a>
Power management		<a href="#">Power management</a>

## 3.7 Analog

### 3.7.1 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 3-23. 16-bit SAR ADC configuration**

**Table 3-34. Reference links to related information**

Topic	Related module	Reference
Full description	16-bit SAR ADC	<a href="#">16-bit SAR ADC</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Signal multiplexing	Port control	<a href="#">Signal multiplexing</a>

### 3.7.1.1 ADC instantiation information

This device contains one ADC.

#### 3.7.1.1.1 Number of ADC channels

The number of ADC channels present on the device is determined by the pinout of the specific device package. For details regarding the number of ADC channel available on a particular package, refer to the signal multiplexing chapter of this MCU.

### 3.7.1.2 DMA Support on ADC

Applications may require continuous sampling of the ADC (4K samples/sec) that may have considerable load on the CPU. Though using PDB to trigger ADC may reduce some CPU load, the ADC supports DMA request functionality for higher performance when the ADC is sampled at a very high rate or cases where PDB is bypassed. The ADC can trigger the DMA (via DMA req) on conversion completion.

### 3.7.1.3 ADCx Connections/Channel Assignment

#### NOTE

As indicated by the following sections, each ADCx\_DP<sub>x</sub> input and certain ADCx\_DM<sub>x</sub> inputs may operate as single-ended ADC channels in single-ended mode.

#### 3.7.1.3.1 ADC0 channel assignment

Table 3-35. ADC0 Assignments

ADC Channel (SC1n[ADCH])	Channel	Input signal (SC1n[DIFF]= 1)	Input signal (SC1n[DIFF]= 0)
00000	DAD0	ADC0_DP0 and ADC0_DM0	ADC0_DP0
00001	DAD1	ADC0_DP1 and ADC0_DM1	ADC0_DP1
00010	DAD2	ADC0_DP2 and ADC0_DM2	ADC0_DP2
00011	DAD3	ADC0_DP3 and ADC0_DM3	ADC0_DP3
00100 <sup>1</sup>	AD4a	Reserved	ADC0_SE4a <sup>2</sup>
00101 <sup>1</sup>	AD5a	Reserved	ADC0_SE5a <sup>3</sup>
00110 <sup>1</sup>	AD6a	Reserved	ADC0_SE6a <sup>4</sup>
00111 <sup>1</sup>	AD7a	Reserved	ADC0_SE7a <sup>5</sup>
00100 <sup>1</sup>	AD4b	Reserved	ADC0_SE4b
00101 <sup>1</sup>	AD5b	Reserved	ADC0_SE5b
00110 <sup>1</sup>	AD6b	Reserved	ADC0_SE6b
00111 <sup>1</sup>	AD7b	Reserved	ADC0_SE7b
01000	AD8	Reserved	ADC0_SE8
01001	AD9	Reserved	ADC0_SE9
01010	AD10	Reserved	Reserved
01011	AD11	Reserved	Reserved
01100	AD12	Reserved	ADC0_SE12
01101	AD13	Reserved	ADC0_SE13
01110	AD14	Reserved	ADC0_SE14
01111	AD15	Reserved	ADC0_SE15
10000	AD16	Reserved	Reserved
10001	AD17	Reserved	ADC0_SE17
10010	AD18	Reserved	ADC0_SE18
10011	AD19	Reserved	ADC0_DM0
10100	AD20	Reserved	ADC0_DM1
10101	AD21	Reserved	Reserved
10110	AD22	Reserved	Reserved
10111	AD23	Reserved	12-bit DAC0 Output/ADC0_SE23
11000	AD24	Reserved	Reserved
11001	AD25	Reserved	Reserved

Table continues on the next page...

Table 3-35. ADC0 Assignments (continued)

ADC Channel (SC1n[ADCH])	Channel	Input signal (SC1n[DIFF]= 1)	Input signal (SC1n[DIFF]= 0)
11010	AD26	Temperature Sensor (Diff)	Temperature Sensor (S.E)
11011	AD27	Bandgap (Diff) <sup>6</sup>	Bandgap (S.E) <sup>6</sup>
11100	AD28	Reserved	Reserved
11101	AD29	-VREFH (Diff)	VREFH (S.E)
11110	AD30	Reserved	VREFL
11111	AD31	Module Disabled	Module Disabled

1. ADCx\_CFG2[MUXSEL] bit selects between ADCx\_SEn channels a and b. Refer to MUXSEL description in ADC chapter for details.
2. Shared device pin with ADC0\_DP1
3. Shared device pin with ADC0\_DM1
4. Shared device pin with ADC0\_DP2
5. Shared device pin with ADC0\_DM2
6. This is the PMC bandgap 1V reference voltage and not the VREF module 1.2 V 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.

### 3.7.1.4 ADC Channels MUX Selection

The following figure shows the assignment of ADCx\_SEn channels a and b through a MUX selection to ADC. To select between alternate set of channels, refer to ADCx\_CFG2[MUXSEL] bit settings for more details.

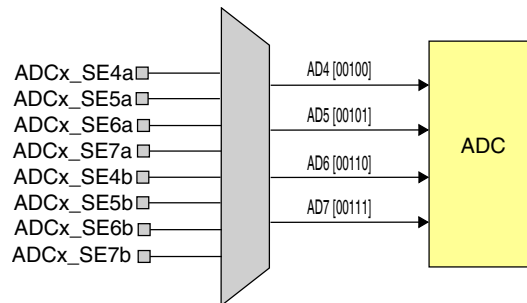


Figure 3-24. ADCx\_SEn channels a and b selection

### 3.7.1.5 ADC Reference Options

The ADC supports the following references:

- VREFH/VREFL - connected as the primary reference option
- 1.2 V VREF\_OUT - connected as the  $V_{ALT}$  reference option

ADCx\_SC2[REFSEL] bit selects the voltage reference sources for ADC. Refer to REFSEL description in ADC chapter for more details.

### NOTE

For this MCU, VREFH shares the VDDA pin in the 32-pin packages. For this MCU, VREFL shares the VSSA pin in the 32-pin packages.

### 3.7.1.6 ADC triggers

The ADC supports both software and hardware triggers. The primary hardware mechanism for triggering the ADC is the PDB. The PDB itself can be triggered by other peripherals. For example: LPTMR Output signal is connected to the PDB. The PDB input trigger can receive the LPTMR Output trigger forcing ADC conversions in run mode (where PDB is enabled). On the other hand, the ADC can conduct conversions in low power modes, not triggered by PDB. This allows the ADC to do conversions in low power mode and store the output in the result register. The ADC generates interrupt when the data is ready in the result register that wakes the system from low power mode. The PDB can also be bypassed by using the ADCxTRGSEL bits in the SIM\_SOPT7 register.

**Table 3-36. ADC Alternate trigger options**

SIM_SOPT7[ADCxTRGSEL]	Selected source
0000	PDB external trigger pin input (PDB0_EXTRG)
0001	CMP0 output
0010	CMP1 output
0011	Reserved
0100	PIT trigger 0
0101	PIT trigger 1
0110	PIT trigger 2
0111	PIT trigger 3
1000	FTM0 trigger
1001	FTM1 trigger
1010	FTM2 trigger
1011	Reserved
1100	Reserved
1101	Reserved
1110	LPTMR trigger
1111	Reserved

For operation of triggers in different modes, refer to Power Management chapter.

### 3.7.1.7 ADC conversion clock options

The ADC has multiple input clock sources. Selection is determined by ADCx\_CFG1[ADICLK] bitfield. The following table shows the chip-specific clock assignments for this bitfield.

#### NOTE

The ALTCLK option is only usable when OSCERCLK is in the MHz range. A system with OSCERCLK in the kHz range has the optional clock source below minimum ADC clock operating frequency.

**Table 3-37. ADC Conversion Clock Options**

ADCx_CFG1[ADICLK]	ADC defined selection	Chip clock	Note
00	Bus Clock	Bus Clock	
01	ALTCLK2	IRC48MCLK	Note <sup>1</sup>
10	ALTCLK	OSCERCLK	Note <sup>1</sup>
11	Asynchronous clock (ADACK)	N/A - sourced from within ADC block	Note <sup>1</sup>

1. For ADC operation in Compute only, PSTOP1, Stop and VLPS, ADACK and the alternate clock sources are allowed clock sources. Note however that ALTCLK2 is force disabled and therefore not available in VLPS.

### 3.7.1.8 ADC low-power modes

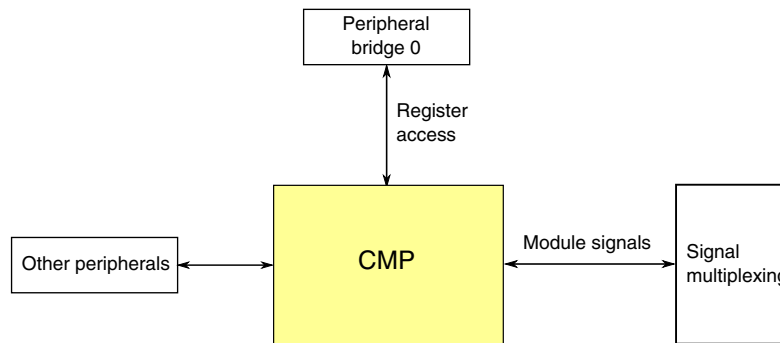
This table shows the ADC low-power modes and the corresponding chip low-power modes.

**Table 3-38. ADC low-power modes**

Module mode	Chip mode
Wait	Wait, VLPW
Normal Stop	Stop, VLPS
Low Power Stop	LLS, VLLS3, VLLS2, VLLS1, VLLS0

## 3.7.2 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 3-25. CMP configuration**

**Table 3-39. Reference links to related information**

Topic	Related module	Reference
Full description	Comparator (CMP)	<a href="#">Comparator</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Signal multiplexing	Port control	<a href="#">Signal multiplexing</a>

### 3.7.2.1 CMP input connections

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

**Table 3-40. CMP input connections**

CMP Inputs	CMP0	CMP1
IN0	CMP0_IN0	CMP1_IN0
IN1	CMP0_IN1	CMP1_IN1
IN2	CMP0_IN2	—
IN3	CMP0_IN3	12-bit DAC0_OUT/CMP1_IN3
IN4	CMP0_IN4	—
IN5	VREF Output/CMP0_IN5	VREF Output/CMP1_IN5
IN6	Bandgap	Bandgap
IN7	6b DAC0 Reference	6b DAC1 Reference

### 3.7.2.2 CMP external references

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

- VREF\_OUT -  $V_{in1}$  input
- VDD -  $V_{in2}$  input

### 3.7.2.3 External window/sample input

PDB pulse-out controls the CMP Sample/Window timing.

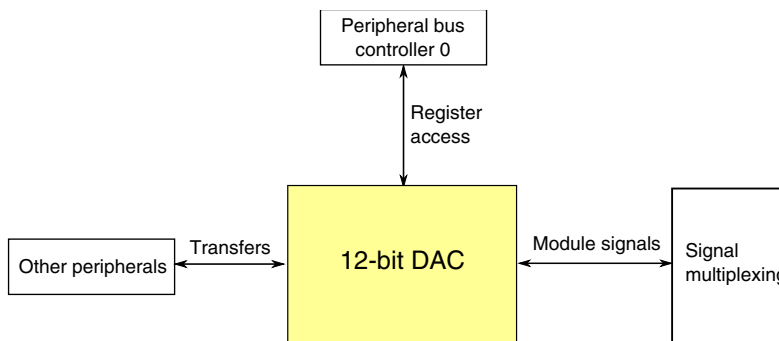
### 3.7.2.4 CMP trigger mode

The CMP and 6-bit DAC sub-block supports trigger mode operation when the  $CMPx\_CR1[TRIGM]$  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 provides a single trigger output to all implemented comparators. Through configuration of the  $CMPx\_CR1[TRIGM]$  bits the trigger can be used to trigger a single comparator or multiple comparators concurrently. 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.

## 3.7.3 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 3-26. 12-bit DAC configuration**

**Table 3-41. Reference links to related information**

Topic	Related module	Reference
Full description	12-bit DAC	<a href="#">12-bit DAC</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Signal multiplexing	Port control	<a href="#">Signal multiplexing</a>

### 3.7.3.1 12-bit DAC Overview

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

### 3.7.3.2 12-bit DAC Output

The output of the DAC can be placed on an external pin or set as one of the inputs to the analog comparator or ADC.

### 3.7.3.3 12-bit DAC Reference

For this device VREF\_OUT and VDDA are selectable as the DAC reference. VREF\_OUT 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 VREF\_OUT reference simultaneously, some degradation of ADC accuracy is to be expected due to DAC switching.



### 3.7.4 VREF 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.

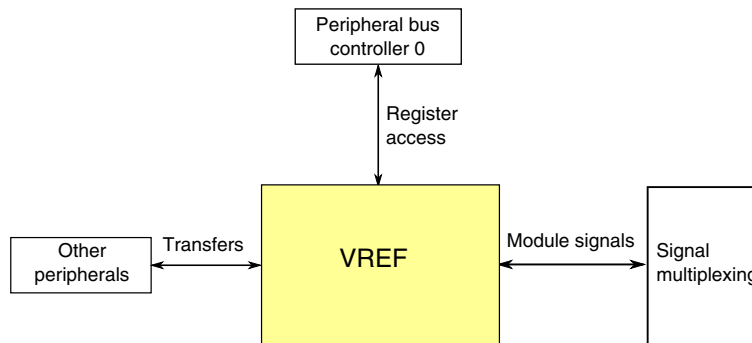


Figure 3-27. VREF configuration

Table 3-42. Reference links to related information

Topic	Related module	Reference
Full description	VREF	<a href="#">VREF</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Signal multiplexing	Port control	<a href="#">Signal multiplexing</a>

#### 3.7.4.1 VREF Overview

This device includes a voltage reference (VREF) to supply an accurate 1.2 V voltage output.

The voltage reference can provide a reference voltage to external peripherals or a reference to analog peripherals, such as the ADC, DAC, or CMP.

#### NOTE

PMC\_REGSC[BGEN] bit must be set if the VREF regulator is required to remain operating in VLPx modes.

#### NOTE

For either an internal or external reference if the VREF\_OUT functionality is being used, VREF\_OUT signal must be

connected to an output load capacitor. Refer the device data sheet for more details.

### 3.8 Timers

#### 3.8.1 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.

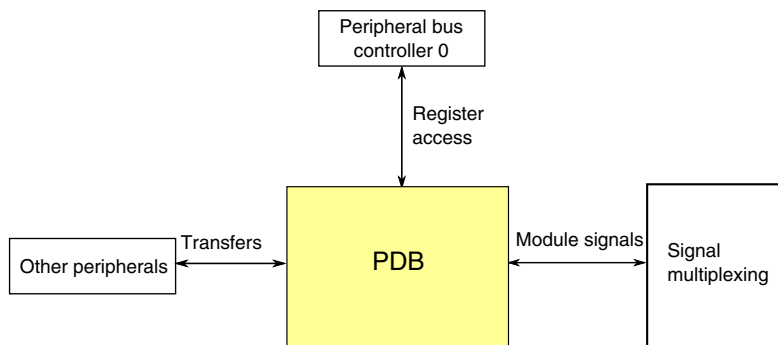


Figure 3-28. PDB configuration

Table 3-43. Reference links to related information

Topic	Related module	Reference
Full description	PDB	<a href="#">PDB</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Signal multiplexing	Port control	<a href="#">Signal multiplexing</a>

#### 3.8.1.1 PDB Instantiation

##### 3.8.1.1.1 PDB Output Triggers

Table 3-44. PDB output triggers

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

Table continues on the next page...

**Table 3-44. PDB output triggers (continued)**

Number of DAC triggers	1
Number of PulseOut	1

### 3.8.1.1.2 PDB Input Trigger Connections

**Table 3-45. PDB Input Trigger Options**

PDB Trigger	PDB Input
0000	External Trigger
0001	CMP 0
0010	CMP 1
0011	Reserved
0100	PIT Ch 0 Output
0101	PIT Ch 1 Output
0110	PIT Ch 2 Output
0111	PIT Ch 3 Output
1000	FTM0 initialization trigger and channel triggers, as programmed in the FTM external trigger register (EXTTRIG)
1001	FTM1 initialization trigger and channel triggers, as programmed in the FTM external trigger register (EXTTRIG)
1010	FTM2 initialization trigger and channel triggers, as programmed in the FTM external trigger register (EXTTRIG)
1011	Reserved
1100	Reserved
1101	Reserved
1110	LPTMR Output
1111	Software Trigger

### 3.8.1.2 PDB Module Interconnections

PDB trigger outputs	Connection
Channel 0 triggers	ADC0 trigger
Channel 1 triggers	synchronous input 1 of FTM0
DAC triggers	DAC0 trigger
Pulse-out	Pulse-out connected to each CMP module's sample/window input to control sample operation

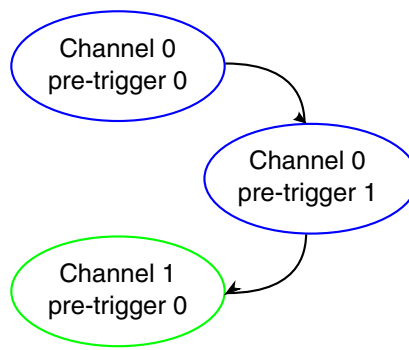
### 3.8.1.3 Back-to-back acknowledgement connections

Back-to-back operation enables the ADC conversions complete to trigger the next PDB channel pre-trigger and trigger output.

In this MCU, PDB back-to-back operation acknowledgment connections are implemented as follows:

- PDB channel 0 trigger/pre-trigger 0 acknowledgement input: None
- PDB channel 0 trigger/pre-trigger 1 acknowledgement input: ADC0SC1A\_COCO
- PDB channel 1 trigger/pre-trigger 0 acknowledgement input: ADC0SC1B\_COCO
- PDB channel 1 trigger/pre-trigger 1 acknowledgement input: None

So, the back-to-back chain is connected as shown below:



**Figure 3-29. PDB back-to-back chain**

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

### 3.8.1.4 PDB Interval Trigger Connections to DAC

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

- PDB interval trigger 0 connects to DAC0 hardware trigger input.

### 3.8.1.5 DAC External Trigger Input Connections

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

- DAC external trigger input 0: ADC0SC1A\_COCO

**NOTE**

Application code can set the PDBx\_DACINTCn[EXT] bit to allow DAC external trigger input when the corresponding ADC Conversion complete flag, ADCx\_SC1n[COCO], is set.

**3.8.1.6 Pulse-Out Connection**

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

**3.8.1.7 Pulse-Out Enable Register Implementation**

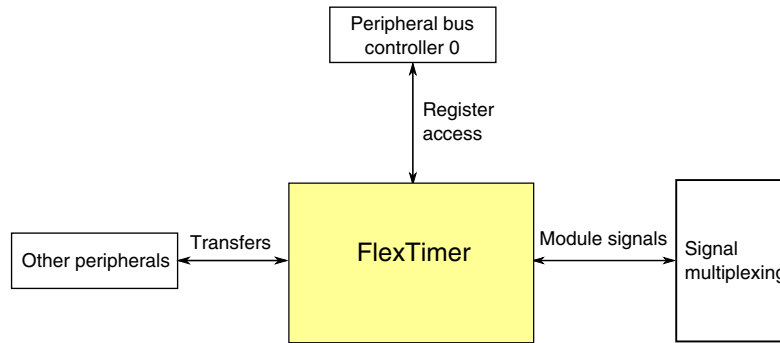
The following table shows the comparison of pulse-out enable register at the module and chip level.

**Table 3-46. 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

**3.8.2 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 3-30. FlexTimer configuration**

**Table 3-47. Reference links to related information**

Topic	Related module	Reference
Full description	FlexTimer	<a href="#">FlexTimer</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock distribution</a>
Power management		<a href="#">Power management</a>
Signal multiplexing	Port control	<a href="#">Signal multiplexing</a>

### 3.8.2.1 Instantiation Information

This device contains three FlexTimer modules.

The following table shows how these modules are configured.

**Table 3-48. FTM Instantiations**

FTM instance	Number of channels	Features/usage
FTM0	6	3-phase motor + 2 general purpose or stepper motor
FTM1	2 <sup>1</sup>	Quadrature decoder or general purpose
FTM2	2 <sup>1</sup>	Quadrature decoder or general purpose

1. Only channels 0 and 1 are available.

### 3.8.2.2 External Clock Options

By default each FTM is clocked by the internal bus 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 two external FTM\_CLKINx pins that can be selected by any FTM module via the SIM\_SOPT4 register.

### 3.8.2.3 Fixed frequency clock

The fixed frequency clock for each FTM is MCGFFCLK.

### 3.8.2.4 FTM Interrupts

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

### 3.8.2.5 FTM Fault Detection Inputs

The following fault detection input options for the FTM modules are selected via the SIM\_SOPT4 register. The external pin option is selected by default.

- FTM0 FAULT0 = FTM0\_FLT0 pin or CMP0 output
- FTM0 FAULT1 = FTM0\_FLT1 pin or CMP1 output
- FTM0 FAULT2 = FTM0\_FLT2 pin
- FTM0 FAULT3 = FTM0\_FLT3 pin
  
- FTM1 FAULT0 = FTM1\_FLT0 pin or CMP0 output
- FTM1 FAULT1 = CMP1 output
  
- FTM2 FAULT0 = FTM2\_FLT0 pin or CMP0 output
- FTM2 FAULT1 = CMP1 output

### 3.8.2.6 FTM Hardware Triggers

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

- FTM0 hardware trigger 0 = SIM\_SOPT8[FTM0SYNCBIT] or CMP0 Output or FTM1 Match (when enabled in the FTM1 External Trigger (EXTTRIG) register)
- FTM0 hardware trigger 1 = PDB channel 1 Trigger Output
- FTM0 hardware trigger 2 = FTM0\_FLT0 pin
  
- FTM1 hardware trigger 0 = SIM\_SOPT8[FTM1SYNCBIT] or CMP0 Output
- FTM1 hardware trigger 1 = CMP1 Output
- FTM1 hardware trigger 2 = FTM1\_FLT0 pin

- FTM2 hardware trigger 0 = SIM\_SOPT8[FTM2SYNCBIT] or CMP0 Output
- FTM2 hardware trigger 2 = FTM2\_FLT0 pin

Having FTMxSYNCBIT fields in the same SOPTx register allows the user to synchronise all FTM timers via their respective TRIG0 input. For the triggers with more than one additional option, the SIM\_SOPT4 register implements control fields for selecting the option.

### 3.8.2.7 Input capture options for FTM module instances

The following channel 0 input capture source options are selected via SIM\_SOPT4. 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
- FTM2 channel 1 input capture = FTM2\_CH1 pin or exclusive OR of FTM2\_CH0, FTM2\_CH1, and FTM1\_CH1. See [FTM Hall sensor support](#).

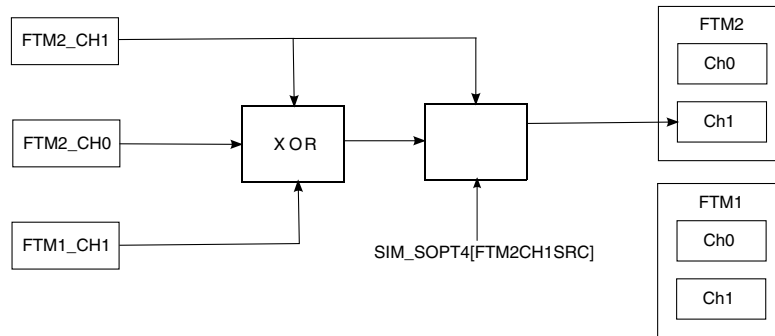
### 3.8.2.8 FTM Hall sensor support

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.

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.





**Figure 3-31. FTM Hall Sensor Configuration**

### 3.8.2.9 FTM modulation implementation

FTM0 support a modulation function where the output channels when configured as PWM or Output Compare mode modulate another timer output when the channel signal is asserted. Any of the 6 channels of FTM0 can be configured to support this modulation function.

The SIM\_SOPT8 register has eight control bits (FTM0CHySRC) that allow the user to select normal PWM/Output Compare mode on the corresponding FTM timer channel or modulate with FTM1\_CH1. The diagram below shows the implementation for FTM0. See SIM Block Guide for further information.

When FTM1\_CH1 is used to modulate an FTM0 channel, then the user must configure FTM1\_CH1 to provide a signal that has a higher frequency than the modulated FTM0 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.

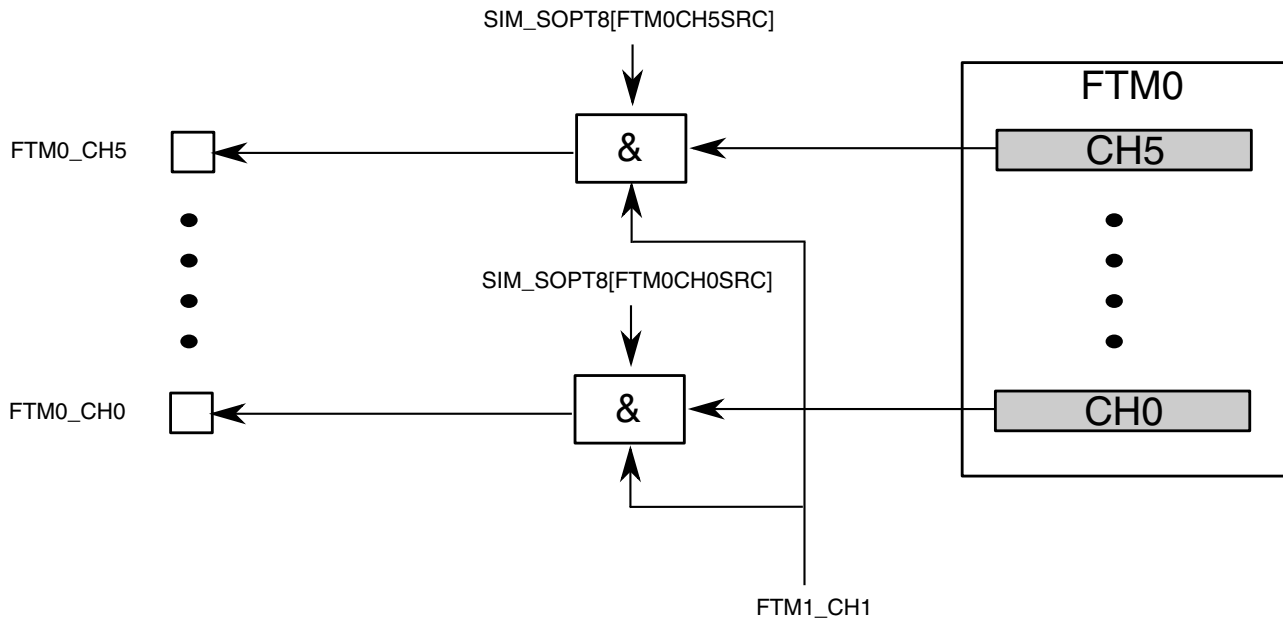


Figure 3-32. FTM Output Modulation

### 3.8.2.10 FTM output triggers for other modules

FTM output triggers can be selected as input triggers for the PDB and ADC modules. See [PDB Instantiation](#) and [ADC triggers](#).

### 3.8.2.11 FTM Global Time Base

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

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

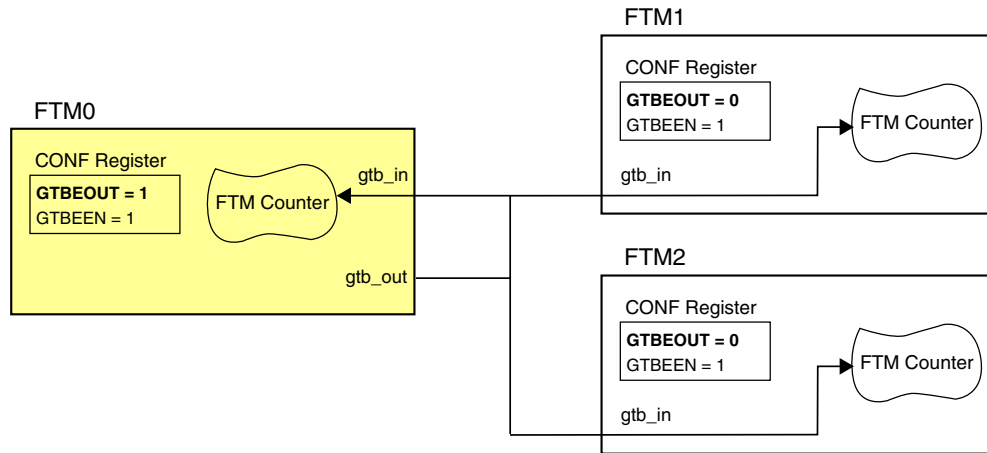


Figure 3-33. FTM Global Time Base Configuration

### 3.8.2.12 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".

### 3.8.3 PIT Configuration

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

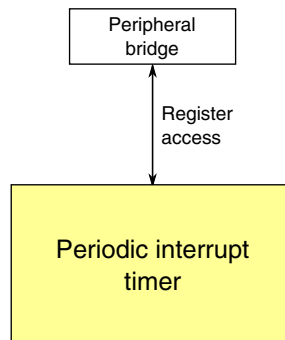


Figure 3-34. PIT configuration

Table 3-49. Reference links to related information

Topic	Related module	Reference
Full description	PIT	<a href="#">PIT</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Power management		<a href="#">Power management</a>

### 3.8.3.1 PIT/DMA Periodic Trigger Assignments

The PIT generates periodic trigger events to the DMA Mux as shown in the table below.

**Table 3-50. PIT channel assignments for periodic DMA triggering**

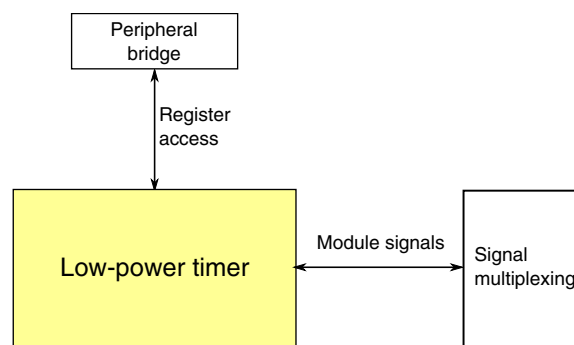
DMA Channel Number	PIT Channel
DMA Channel 0	PIT Channel 0
DMA Channel 1	PIT Channel 1
DMA Channel 2	PIT Channel 2
DMA Channel 3	PIT Channel 3

### 3.8.3.2 PIT/ADC Triggers

PIT triggers are selected as ADCx trigger sources using the SIM\_SOPT7[ADCxTRGSEL] fields. For more details, refer to SIM chapter.

## 3.8.4 Low-power timer 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. LPTMR configuration**

**Table 3-51. Reference links to related information**

Topic	Related module	Reference
Full description	Low-power timer	<a href="#">Low-power timer</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Power management		<a href="#">Power management</a>

*Table continues on the next page...*

**Table 3-51. Reference links to related information (continued)**

Topic	Related module	Reference
Signal Multiplexing	Port control	<a href="#">Signal Multiplexing</a>

### 3.8.4.1 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 VLPS/LLS/VLLS modes)
01	1	LPO — 1 kHz clock (not available in VLLS0 mode)
10	2	ERCLK32K — secondary external reference clock
11	3	OSCERCLK_UNDIV — Undivided external reference clock (not available in VLLS0 mode)

See [Clock Distribution](#) for more details on these clocks.

### 3.8.4.2 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	LPTMR_ALT1 pin
10	2	LPTMR_ALT2 pin
11	3	LPTMR_ALT3 pin

## 3.9 Communication interfaces

### 3.9.1 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.

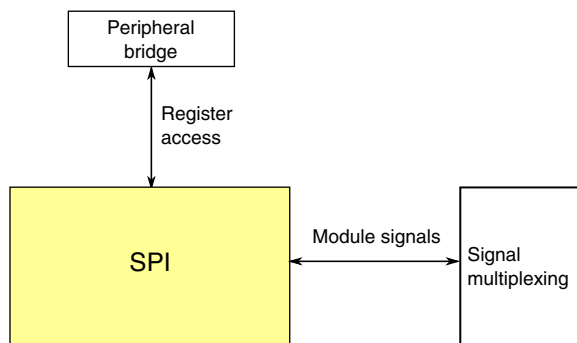


Figure 3-36. SPI configuration

Table 3-52. Reference links to related information

Topic	Related module	Reference
Full description	SPI	<a href="#">SPI</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Signal Multiplexing	Port control	<a href="#">Signal Multiplexing</a>

#### 3.9.1.1 SPI Modules Configuration

This device contains one SPI module .

#### 3.9.1.2 SPI clocking

The SPI module is clocked by the internal bus clock (the DSPI refers to it as system clock). The module has an internal divider, with a minimum divide is two. So, the SPI can run at a maximum frequency of bus clock/2.

### 3.9.1.3 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 all 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.

### 3.9.1.4 TX FIFO size

**Table 3-53. SPI transmit FIFO size**

SPI Module	Transmit FIFO size
SPI0	4

### 3.9.1.5 RX FIFO Size

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

**Table 3-54. SPI receive FIFO size**

SPI Module	Receive FIFO size
SPI0	4

### 3.9.1.6 Number of PCS signals

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

**Table 3-55. SPI PCS signals**

SPI Module	PCS Signals
SPI0	For packages with 64 pins: SPI0_PCS[4:0] For packages with 48 pins: SPI0_PCS[4:0] For packages with 32 pins: SPI0_PCS[3:0]

### 3.9.1.7 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 2MHz.

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.

#### 3.9.1.7.1 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.

### 3.9.1.8 SPI Doze Mode

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

### 3.9.1.9 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.



### 3.9.1.10 SPI clocks

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

**Table 3-56. SPI clock connections**

Module clock	Chip clock
System Clock	Bus Clock

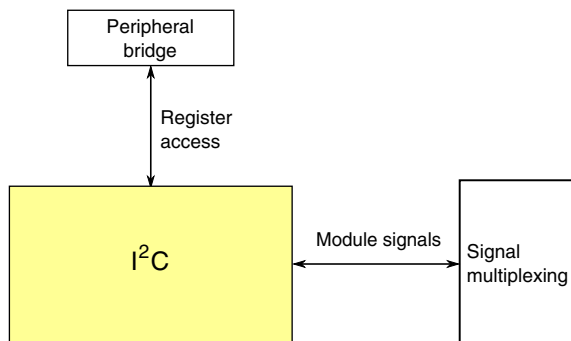
### 3.9.1.11 Writing SPI Transmit FIFO

The SPI supports 8-bit or 16-bit writes to the PUSH TX FIFO, allowing a single write to the command word followed by multiple writes to the transmit word. The TX FIFO will save the last command word written, and convert a 8-bit/16-bit write to the transmit word into a 32-bit write that pushes both the command word and transmit word into the TX FIFO ([PUSH TX FIFO Register In Master Mode](#))

A 32-bit write to the SPI\_PUSH register will push all 32-bits to the TX FIFO. An 8-bit or 16-bit write to the 16-bit transmit data field will push the data together with the last written command word. An 8-bit or 16-bit write to the command word does not push data onto the FIFO, but that command word is pushed to the TX FIFO on all subsequent 8-bit or 16-bit writes to the transmit data field. This allows a single 16-bit write to the command word to be used for all subsequent 8-bit or 16-bit writes to the transmit data word. Writing a different 16-bit command word will cause all subsequent 8-bit or 16-bit writes to the transmit data word to be pushed to the TX FIFO with the new command word.

## 3.9.2 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 3-37. I2C configuration**

**Table 3-57. Reference links to related information**

Topic	Related module	Reference
Full description	I2C	<a href="#">I2C</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Power management		<a href="#">Power management</a>
Signal Multiplexing	Port control	<a href="#">Signal Multiplexing</a>

### 3.9.2.1 I2C Instantiation Information

This device has one I<sup>2</sup>C modules.

The I2C module includes SMBus support and DMA support. It also has optional address match wakeup in Stop/VLPS mode.

The digital glitch filter implemented in the IIC module, controlled by the I2Cx\_FLT[FLT] registers, is clocked from the bus clock and thus has filter granularity in bus clock cycle counts.

### 3.9.3 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.

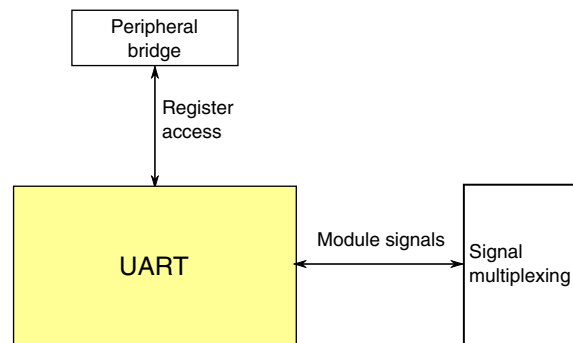


Figure 3-38. UART configuration

Table 3-58. Reference links to related information

Topic	Related module	Reference
Full description	UART	<a href="#">UART</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Power management		<a href="#">Power management</a>
Signal Multiplexing	Port control	<a href="#">Signal Multiplexing</a>

### 3.9.3.1 UART configuration information

This section describes how each module is configured on this device.

- 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
- UART0 and UART1 are clocked from the core clock. The maximum baud rate is 1/16 of related source clock frequency.
- IrDA is available on all UARTs
- UART0 implements 8-entry transmit and 8-entry receive FIFOs
- UART1 implements 1-entry transmit and 1-entry receive FIFOs

### 3.9.3.2 UART wakeup

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

### 3.9.3.3 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

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 Human-machine interfaces

### 3.10.1 GPIO configuration

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

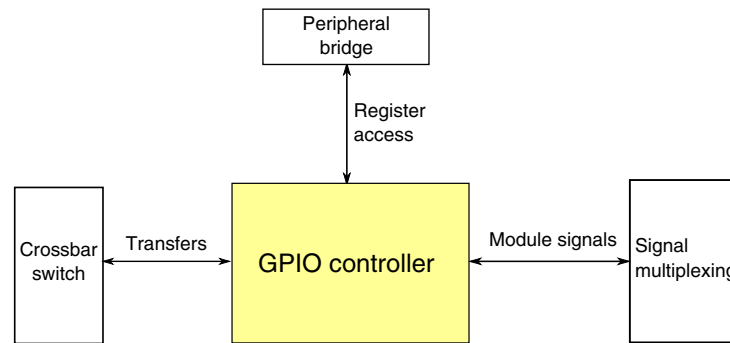


Figure 3-39. GPIO configuration

Table 3-59. Reference links to related information

Topic	Related module	Reference
Full description	GPIO	<a href="#">GPIO</a>
System memory map		<a href="#">System memory map</a>
Clocking		<a href="#">Clock Distribution</a>
Power management		<a href="#">Power management</a>
Transfers	Crossbar switch	<a href="#">Clock Distribution</a>
Signal Multiplexing	Port control	<a href="#">Signal Multiplexing</a>

### 3.10.1.1 Number of GPIO signals

The number of GPIO signals available on the devices covered by this document are detailed in [Orderable part numbers](#) .

Eight GPIO pins support a high drive capability - PTB0, PTB1, PTD4, PTD5, PTD6, PTD7, PTC3, and PTC4. All other GPIO support normal drive option only.

PTA4 includes a passive input filter that is enabled or disabled by PORTA\_PCR4[PFE] control. This reset default is to have this function disabled.



# Chapter 4

## Memory Map

### 4.1 Introduction

This device contains various memories and memory-mapped peripherals which are located in one 32-bit contiguous 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. This map provides the complete architectural address space definition for the various sections. Based on the physical sizes of the memories and peripherals, the actual address regions used may be smaller.

The system memory map includes address spaces that are intended for specific purposes.

- There is an aliased region that maps a system address space to the Program flash section. Flash region aliasing is specifically intended for references to read-only data coefficients in the flash while still preserving a full Harvard memory organization in the processor core supporting concurrent instruction fetches (for example, from RAM) and data accesses (from flash via the aliased space).
- The bitbanding functionality supported by the processor core uses aliased regions that map to the basic RAM and peripheral address spaces. This functionality maps each 32-bit word of the aliased address space to a unique bit in the underlying RAM or peripheral address space to support single-bit insert and extract operations from the processor.

**Table 4-1. System memory map**

System 32-bit Address Range	Destination Slave	Access
0x0000_0000–0x07FF_FFFF <sup>1</sup>	Program flash and read-only data (Includes exception vectors in first 1024 bytes)	All masters
0x0800_0000–0x0FFF_FFFF	Reserved	—
0x1000_0000–0x1BFF_FFFF	Reserved	—
0x1C00_0000–0x1FFF_FFFF <sup>2</sup>	SRAM_L: Lower SRAM (ICODE/DCODE)	All masters
0x2000_0000–0x200F_FFFF <sup>2</sup>	SRAM_U: Upper SRAM bitband region	All masters
0x2010_0000–0x21FF_FFFF	Reserved	—
0x2200_0000–0x23FF_FFFF	Aliased to SRAM_U bitband	Cortex-M4 core only
0x2400_0000–0x2FFF_FFFF	Reserved	—
0x3000_0000–0x33FF_FFFF <sup>1</sup>	Program Flash and read-only data	Cortex-M4 core only
0x3400_0000–0x3FFF_FFFF	Reserved	—
0x4000_0000–0x4007_FFFF	Bitband region for peripheral bridge 0 (AIPS-Lite0)	Cortex-M4 core & DMA
0x4008_0000–0x400F_EFFF	Reserved	—
0x400F_F000–0x400F_FFFF	Bitband region for general purpose input/output (GPIO)	Cortex-M4 core & DMA
0x4010_0000–0x41FF_FFFF	Reserved	—
0x4200_0000–0x42FF_FFFF	Aliased to peripheral bridge (AIPS-Lite) bitband	Cortex-M4 core only
0x4300_0000–0x43FD_FFFF	Reserved	—
0x43FE_0000–0x43FF_FFFF	Aliased to general purpose input/output (GPIO) bitband	Cortex-M4 core only
0x4400_0000–0xDFFF_FFFF	Reserved	—
0xE000_0000–0xE00F_FFFF	Private peripherals	Cortex-M4 core only
0xE010_0000–0xFFFF_FFFF	Reserved	—

1. This map provides the complete architectural address space definition for the flash. Based on the physical sizes of the memories implemented for a particular device, the actual address regions used may be smaller. See [Flash Memory Sizes](#) for details.
2. This range varies depending on amount of SRAM implemented for a particular device. See [SRAM sizes](#) for details.

### NOTE

1. Access rights to AIPS-Lite peripheral bridge and general purpose input/output (GPIO) module address space is limited to the core, DMA .
2. ARM Cortex-M4 core access privileges also includes accesses via the debug interface.



## 4.2.1 Aliased bit-band regions

The SRAM\_U, AIPS-Lite, and general purpose input/output (GPIO) module resources reside in the Cortex-M4 processor bit-band regions.

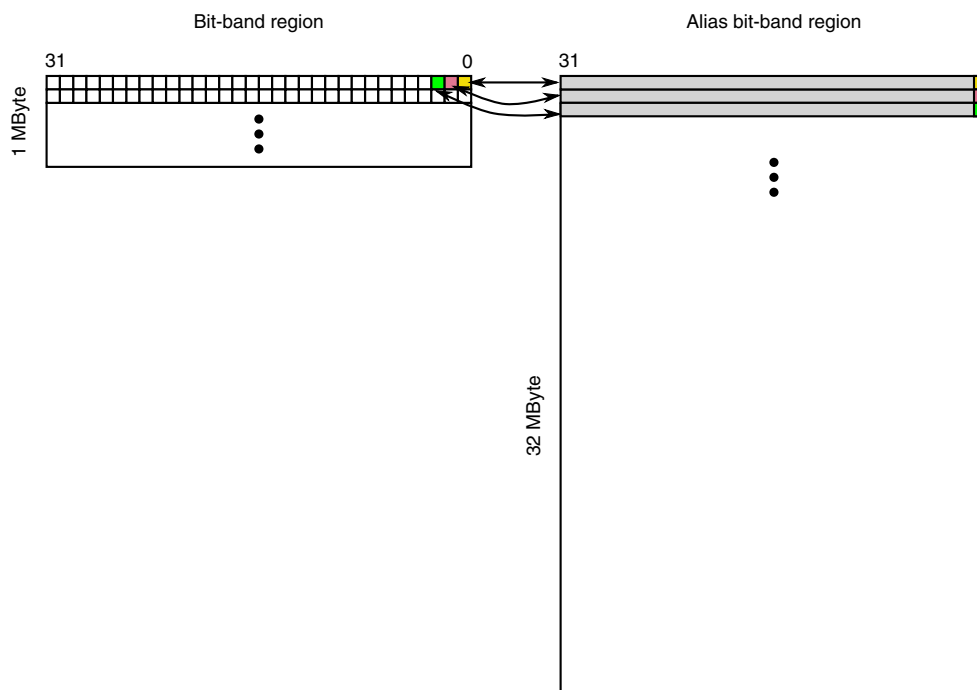
The processor also includes two 32 MB aliased bit-band regions associated with the two 1 MB bit-band spaces. Each 32-bit location in the 32 MB space maps to an individual bit in the bit-band region. A 32-bit write in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

Bit 0 of the value written to the alias region determines what value is written to the target bit:

- Writing a value with bit 0 set writes a 1 to the target bit.
- Writing a value with bit 0 clear writes a 0 to the target bit.

A 32-bit read in the alias region returns either:

- a value of 0x0000\_0000 to indicate the target bit is clear
- a value of 0x0000\_0001 to indicate the target bit is set



**Figure 4-1. Alias bit-band mapping**

### NOTE

Each bit in bit-band region has an equivalent bit that can be manipulated through bit 0 in a corresponding long word in the alias bit-band region.

## 4.2.2 Flash Access Control Introduction

The Flash Access Control (FAC) is a Freescale or third-party configurable memory protection scheme optimized to allow end users to utilize software libraries while offering programmable restrictions to these libraries. The flash memory is divided into equal size segments that provide protection to proprietary software libraries. The protection of these segments is controlled as the FAC provides a cycle-by-cycle evaluation of the access rights for each transaction routed to the on-chip flash memory. Configurability allows an increasing number of protected segments while supporting two levels of vendors adding their proprietary software to a device.

Flash access control aligns to the three privilege levels supported by ARM Cortex-M family products where the most secure state - supervisor/privileged secure - aligns to the execute-only and supervisor-only access control. The unsecure state of user non-secure aligns to no access control states set, and the mid-level state where user secure aligns to using the access control of execute-only.

Control for this protection scheme is implemented in Program Once NVM locations and is configurable through a Program Once flash command operations. The NVM locations controlling FAC are unaffected by Erase All Blocks flash command and debug interface initiated mass erase operations.

### NOTE

The FAC protection scheme has eight XACC and eight SACC registers to control up to 64 segments. For program flash sizes 128KB or less, the memory is divided into 32 segments, controlled by the four lower-order XACC and SACC registers.

## 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](#).

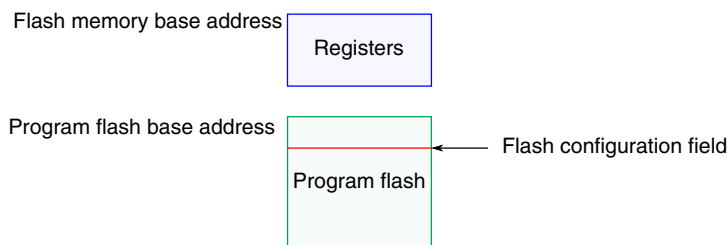


Figure 4-2. Flash memory map

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 6)	FCFTRIM
0x0000_03FF	SCTRIM

## 4.4 SRAM memory map

The on-chip RAM is split in two regions: SRAM\_L and SRAM\_U. The RAM is implemented such that the SRAM\_L and SRAM\_U ranges form a contiguous block in the memory map. See [SRAM Configuration](#) 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 Peripheral bridge (AIPS-Lite) memory map

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.

### 4.5.1 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.

#### NOTE

One factor contributing to these situations is processor write buffering. The processor architecture has a programmable configuration bit to disable write buffering: ACTLR[DISDEFWBUF]. However, disabling buffered writes is likely to degrade system performance much more than simply performing the required memory serialization for the situations that truly require it.

### 4.5.2 Peripheral Bridge 0 (AIPS-Lite 0) 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	—

*Table continues on the next page...*

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

System 32-bit base address	Slot number	Module
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	—
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	Flash memory controller
0x4002_0000	32	Flash memory
0x4002_1000	33	DMA channel mutiplexer
0x4002_2000	34	—
0x4002_3000	35	—
0x4002_4000	36	—
0x4002_5000	37	—
0x4002_6000	38	—
0x4002_7000	39	—
0x4002_8000	40	—
0x4002_9000	41	—

*Table continues on the next page...*

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

System 32-bit base address	Slot number	Module
0x4002_A000	42	—
0x4002_B000	43	—
0x4002_C000	44	SPI 0
0x4002_D000	45	—
0x4002_E000	46	—
0x4002_F000	47	—
0x4003_0000	48	—
0x4003_1000	49	—
0x4003_2000	50	CRC
0x4003_3000	51	—
0x4003_4000	52	—
0x4003_5000	53	—
0x4003_6000	54	Programmable delay block (PDB)
0x4003_7000	55	Periodic interrupt timers (PIT)
0x4003_8000	56	FlexTimer (FTM) 0
0x4003_9000	57	FlexTimer (FTM) 1
0x4003_A000	58	FlexTimer (FTM) 2
0x4003_B000	59	Analog-to-digital converter (ADC) 0
0x4003_C000	60	—
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
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	—

*Table continues on the next page...*

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

System 32-bit base address	Slot number	Module
0x4005_1000	81	—
0x4005_2000	82	Software watchdog
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
0x4006_2000	98	—
0x4006_3000	99	—
0x4006_4000	100	Multi-purpose Clock Generator (MCG)
0x4006_5000	101	System oscillator (OSC)
0x4006_6000	102	I <sup>2</sup> C 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	—
0x4007_1000	113	—
0x4007_2000	114	—
0x4007_3000	115	Analog comparator (CMP) / 6-bit digital-to-analog converter (DAC)
0x4007_4000	116	Voltage reference (VREF)
0x4007_5000	117	—
0x4007_6000	118	—
0x4007_7000	119	—

*Table continues on the next page...*

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

System 32-bit base address	Slot number	Module
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		GPIO controller

## 4.6 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
0xE000_0000–0xE000_0FFF	Instrumentation Trace Macrocell (ITM)
0xE000_1000–0xE000_1FFF	Data Watchpoint and Trace (DWT)
0xE000_2000–0xE000_2FFF	Flash Patch and Breakpoint (FPB)
0xE000_3000–0xE000_DFFF	Reserved
0xE000_E000–0xE000_EFFF	System Control Space (SCS) (for NVIC and FPU)
0xE000_F000–0xE003_FFFF	Reserved
0xE004_0000–0xE004_0FFF	Trace Port Interface Unit (TPIU)
0xE004_1000–0xE004_1FFF	Reserved
0xE004_2000–0xE004_2FFF	Reserved
0xE004_3000–0xE004_3FFF	Reserved
0xE004_4000–0xE007_FFFF	Reserved
0xE008_0000–0xE008_0FFF	Miscellaneous Control Module (MCM)
0xE008_1000–0xE008_1FFF	Reserved
0xE008_2000–0xE00F_EFFF	Reserved
0xE00F_F000–0xE00F_FFFF	ROM Table - allows auto-detection of debug components



# Chapter 5

## Clock Distribution

### 5.1 Introduction

The MCG module controls which clock source is used to derive the system clocks. The clock generation logic divides the selected clock source into a variety of clock domains, including the clocks for the system bus masters, system bus slaves, and flash memory . The clock generation logic also implements module-specific clock gating to allow granular shutoff of modules.

The primary clocks for the system are generated from the MCGOUTCLK clock. The clock generation circuitry provides several clock dividers that allow different portions of the device to be clocked at different frequencies. This allows for trade-offs between performance and power dissipation.

Various modules have module-specific clocks that can be generated from the IRC48MCLK or MCGFLLCLK clock. In addition, there are various other module-specific clocks that have other alternate sources. Clock selection for most modules is controlled by the SOPT registers in the SIM module.

### 5.2 Programming model

The selection and multiplexing of system clock sources is controlled and programmed via the MCG module. The setting of clock dividers and module clock gating for the system are programmed via the SIM module. Reference 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 below figure:

## Clock definitions

	OSC	MCG	SIM
Multiplexers	MCG_Cx	MCG_Cx	SIM_SOPT1, SIM_SOPT2
Dividers	—	MCG_Cx	SIM_CLKDIVx
Clock gates	OSC_CR	MCG_C1	SIM_SCGCx

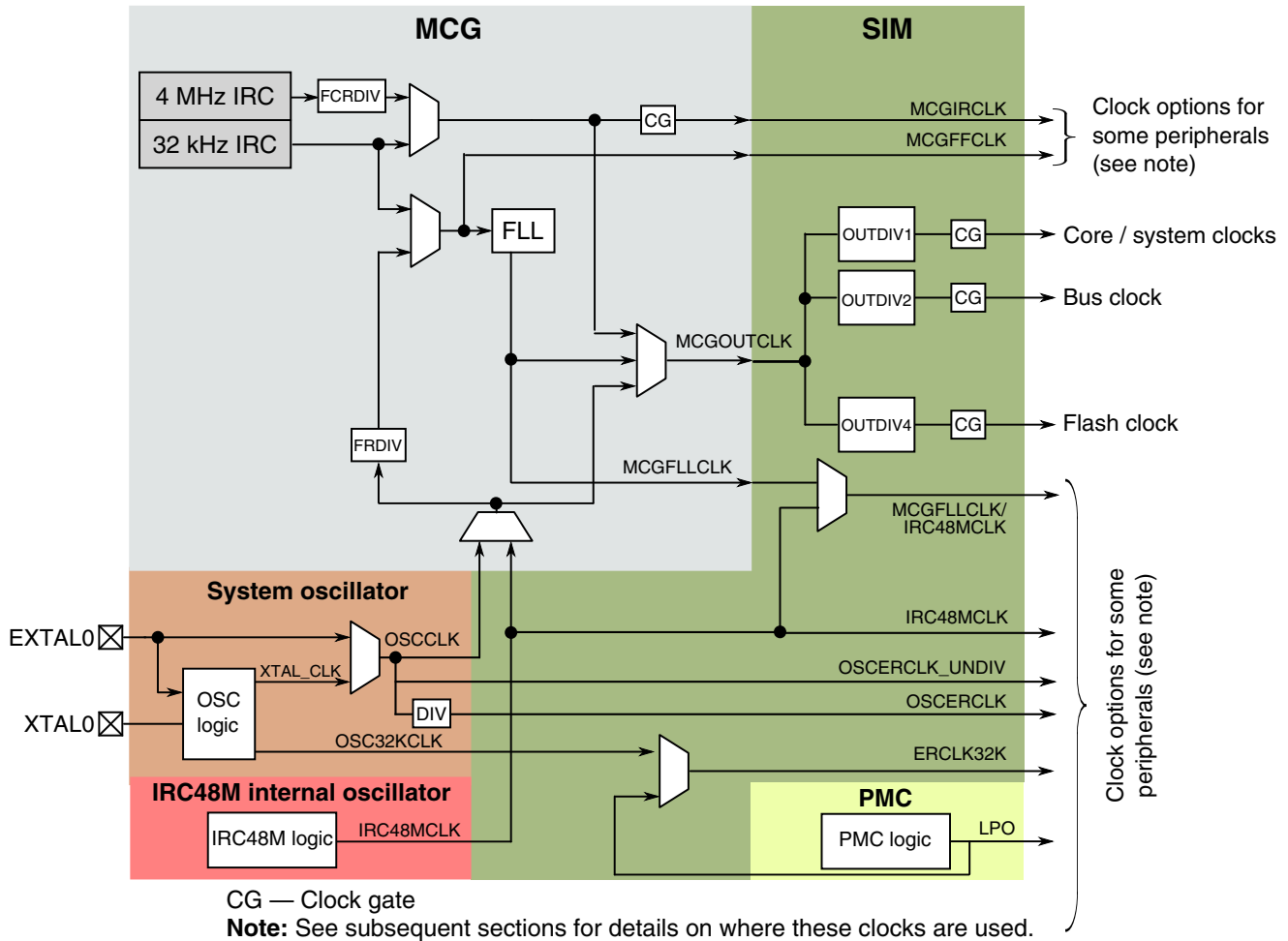


Figure 5-1. Clocking diagram

## 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-M4 core

Table continues on the next page...

Clock name	Description
Platform clock	MCGOUTCLK divided by OUTDIV1, clocks the crossbar switch and NVIC.
System clock	MCGOUTCLK divided by OUTDIV1, clocks the bus masters directly. In addition, this clock is used for UART0 and UART1.
Bus clock	MCGOUTCLK divided by OUTDIV2 clocks the bus slaves and peripheral (excluding memories)
Flash clock	MCGOUTCLK divided by OUTDIV4 clocks the flash memory
MCGIRCLK	MCG output of the slow or fast internal reference clock
MCGFFCLK	MCG output of the slow internal reference clock or a divided MCG external reference clock.
MCGOUTCLK	MCG output of either IRC, MCGFLLCLK or MCG's external reference clock that sources the core, system, bus, and flash clock. It is also an option for the debug trace clock.
MCGFLLCLK	MCG output of the FLL. MCGFLLCLK may clock some modules.
IRC48MCLK	Internal 48 MHz oscillator that can be used as a reference to the MCG and also may clock some on-chip modules.
OSCCLK	System oscillator output of the internal oscillator or sourced directly from EXTAL
OSCCERCLK	System oscillator output sourced from OSCCLK that may clock some on-chip modules. Dividable by 1, 2, 4, or 8.
OSC32KCLK	System oscillator 32kHz output
ERCLK32K	Clock source for some modules that is chosen as OSC32KCLK.
LPO	PMC 1kHz output

## 5.4.1 Device clock summary

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

**Table 5-1. Clock Summary**

Clock name	High Speed Run mode clock frequency	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when...
MCGOUTCLK	Up to 100 MHz	Up to 100 MHz	Up to 4 MHz	MCG	In all stop modes except for partial stop modes.
MCGFLLCLK	Up to 100 MHz	Up to 100 MHz	N/A	MCG	MCG clock controls do not enable. Overriding forced disable in all low powers modes (including STOP and VLPx modes).

*Table continues on the next page...*

Table 5-1. Clock Summary (continued)

Clock name	High Speed Run mode clock frequency	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when...
Core clock	Up to 100 MHz	Up to 72 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all wait and stop modes
System clock	Up to 100 MHz	Up to 72 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all stop modes and Compute Operation
Bus clock	Up to 50 MHz	Up to 50 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all stop modes except for partial STOP2 mode, and Compute Operation
Flash clock	Up to 25 MHz	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 or 4 MHz	30-40 kHz or 4 MHz	4 MHz only	MCG	MCG_C1[IRCLKEN] cleared, Stop or VLPS mode and MCG_C1[IREFSTE N] cleared, or LLS/VLLS mode
External reference (OSCERCLK)	Up to 50 MHz (bypass), 30-40 kHz, or 3-32 MHz (crystal)	Up to 50 MHz (bypass), 30-40 kHz, or 3-32 MHz (crystal)	Up to 16 MHz (bypass), 30-40 kHz (low-range crystal) or Up to 16 MHz (high-range crystal)	System OSC	System OSC's OSC_CR[ERCLKEN] cleared, or Stop mode and OSC_CR[EREFSTEN] cleared
External reference 32kHz (ERCLK32K)	30-40 kHz	30-40 kHz	30-40 kHz	System OSC or LPO	System OSC's OSC_CR[ERCLKEN] cleared
Internal 48 MHz clock (IRC48MCLK)	48 MHz	48 MHz	N/A	IRC48M	MCG or SIM control does not enable. Overriding forced disable in VLPS, LLSx, VLLSx.
CLKOUT32K	32 kHz	32 kHz	32 kHz	ERCLK32K - which is system OSC or LPO depending on SIM_SOPT1[OSC32KSEL]	SIM_SOPT1[OSC32KOUT] not configured to drive ERCLK32K out.
LPO	1 kHz	1 kHz	1 kHz	PMC	in VLLS0
TRACE clock	Up to 100 MHz	Up to 100 MHz	Up to 4 MHz	System clock or MCGOUTCLK	Trace is disabled

## 5.5 Internal clocking requirements

The clock dividers are programmed via the SIM module's CLKDIV registers. Each divider is programmable from a divide-by-1 through divide-by-16 setting. The following requirements must be met when configuring the clocks for this device:

1. The core and system clock frequencies must be 100 MHz or slower in HSRUN, 72 MHz or slower in RUN.
2. The bus clock frequency must be programmed to 50 MHz or less in both HSRUN and RUN, and an integer divide of the core clock. The core clock to bus clock ratio is limited to a max value of 8.
3. The flash clock frequency must be programmed to 25 MHz or less, less than or equal to the bus clock, and an integer divide of the core clock. The core clock to flash clock ratio is limited to a max value of 8.

The following are a few of the more common clock configurations for this device:

### Option 1:

<b>Clock</b>	<b>Frequency</b>
Core clock	50 MHz
System clock	50 MHz
Bus clock	50 MHz
Flash clock	25 MHz

### Option 2: Run

<b>Clock</b>	<b>Frequency</b>
Core clock	72 MHz
System clock	72 MHz
Bus clock	36 MHz
Flash clock	24 MHz

### Option 3: High Speed Run

<b>Clock</b>	<b>Frequency</b>
Core clock	100 MHz
System clock	100 MHz
Bus clock	50 MHz
Flash clock	25 MHz

### 5.5.1 Clock divider values after reset

Each clock divider is programmed via the SIM module's CLKDIV $n$  registers. The flash memory's FTF\_FOPT[LPBOOT] bit controls the reset value of the core clock, system clock, bus clock, and flash clock dividers as shown below:

FTF_FOPT [LPBOOT]	Core/system clock	Bus clock	Flash clock	Description
0	0x7 (divide by 8)	0x7 (divide by 8)	0xF (divide by 16)	Low power boot
1	0x0 (divide by 1)	0x0 (divide by 1)	0x1 (divide by 2)	Fast clock boot

This gives the user flexibility for a lower frequency, low-power boot option. The flash erased state defaults to fast clocking mode, since where the low power boot (FTF\_FOPT[LPBOOT]) bit resides in flash is logic 1 in the flash erased state.

To enable the low power boot option program FTF\_FOPT[LPBOOT] to zero. During the reset sequence, if LPBOOT is cleared, the system is in a slow clock configuration. Upon 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:

- the core/system and bus clocks are less than or equal to 4 MHz, and
- the flash memory clock is 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] and SIM\_CLKDIV1[OUTDIV4], must be programmed such 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 and SIM\_CLKDIV1[OUTDIV4]=0100b, 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 SCGCx 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 SCGCx register to enable the clock. Before turning off the clock, make sure to disable the module.

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
<b>Core modules</b>			
ARM Cortex-M4 core	System clock	Core clock	—
NVIC	System clock	—	—
DAP	System clock	—	—
ITM	System clock	—	—
cJTAG, JTAGC	—	—	JTAG_CLK
<b>System modules</b>			
DMA	System clock	—	—
DMA Mux	Bus clock	—	—
Port control	Bus clock	LPO	—
Crossbar Switch	System clock	—	—
Peripheral bridges	System clock	Bus clock, Flash clock	—
LLWU, PMC, SIM, RCM	Flash clock	LPO	—
Mode controller	Flash clock	—	—
MCM	System clock	—	—
EWM	Bus clock	LPO	—
Watchdog timer	Bus clock	LPO	—
<b>Clocks</b>			
MCG	Flash clock	MCGOUTCLK, MCGFLLCLK, MCGIRCLK, OSCCLK, IRC48MCLK	—
OSC	Bus clock	OSCCERCLK, OSCCLK, OSCCERCLK_UNDIV, OSC32KCLK	—

*Table continues on the next page...*

**Table 5-2. Module clocks (continued)**

Module	Bus interface clock	Internal clocks	I/O interface clocks
IRC48M	—	IRC48MCLK	—
<b>Memory and memory interfaces</b>			
Flash Controller	System clock	Flash clock	—
Flash memory	Flash clock	—	—
<b>Security</b>			
CRC	Bus clock	—	—
<b>Analog</b>			
ADC	Bus clock	OSCERCLK , IRC48MCLK	—
CMP	Bus clock	—	—
DAC	Bus clock	—	—
VREF	Flash clock	—	—
<b>Timers</b>			
PDB	Bus clock	—	—
FlexTimers	Bus clock	MCGFFCLK	FTM_CLKINx
PIT	Bus clock	—	—
LPTMR	Flash clock	LPO, OSCERCLK_UNDIV, MCGIRCLK, ERCLK32K	—
<b>Communication interfaces</b>			
DSPI	Bus clock	—	DSPI_SCK
I <sup>2</sup> C	Bus clock	—	I2C_SCL
UART0, UART1	System clock	—	—
<b>Human-machine interfaces</b>			
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 LPO clock or 1-kHz LPO clock.

### 5.7.2 IRC 48MHz clock

The integrated 48 MHz internal reference clock source (IRC48MCLK) is available in High Speed Run, Run, WAIT and Stop modes of operation. IRC48MCLK is also available in Compute Only, PSTOP2 and PSTOP1 modes of operation when entered from Run mode. IRC48MCLK is forced disabled when the MCU transitions into VLPS, LLSx, and VLLSx low power modes.



**NOTE**

IRC48MCLK is not forced disabled in Stop modes and should be disabled by software prior to Stop entry unless it is required. IRC48MCLK is not forced disabled in VLPR and should be disabled by software prior to VLPR entry.

IRC48MCLK is enabled via any of the following control settings while operating in these modes:

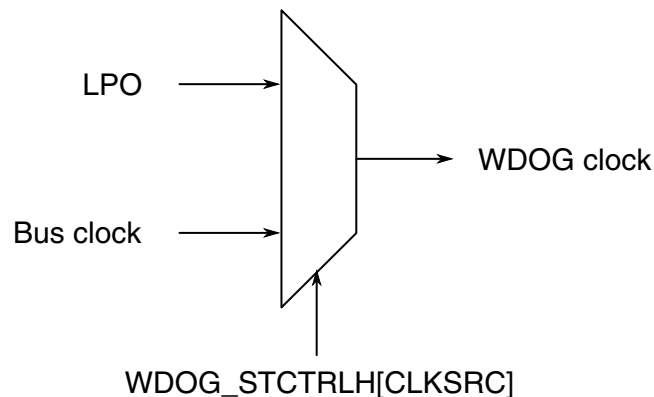
- MCG Control register selects IRC48 MHz clock (enabled when MCG\_C7[OSCSSEL]=10) *and* MCG is configured in an external clocking mode (FBE or FEE).
- SIM Control register selects IRC48 MHz clock — enabled when SIM\_SOPT2[PLLFLSEL]=11

The IRC48MCLK is available for use as:

- an oscillator reference to the MCG - from which core, system, bus, and flash clock sources can be derived
- an ADC alternate clock source

### 5.7.3 WDOG clocking

The WDOG may be clocked from two clock sources as shown in the following figure.



**Figure 5-2. WDOG clock generation**

### 5.7.4 Debug trace clock

The debug trace clock source can be clocked as shown in the following figure.

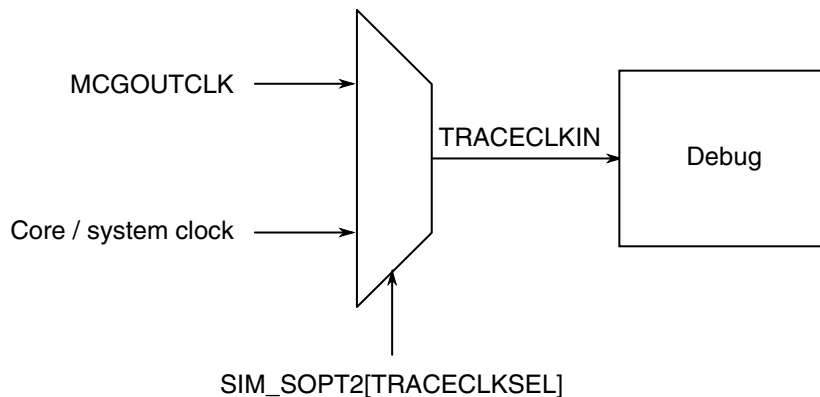


Figure 5-3. Trace clock generation

### 5.7.5 PORT digital filter clocking

The digital filters in the PORTD module can be clocked as shown in the following figure.

**NOTE**

In stop mode, the digital input filters are bypassed unless they are configured to run from the 1 kHz LPO clock source.

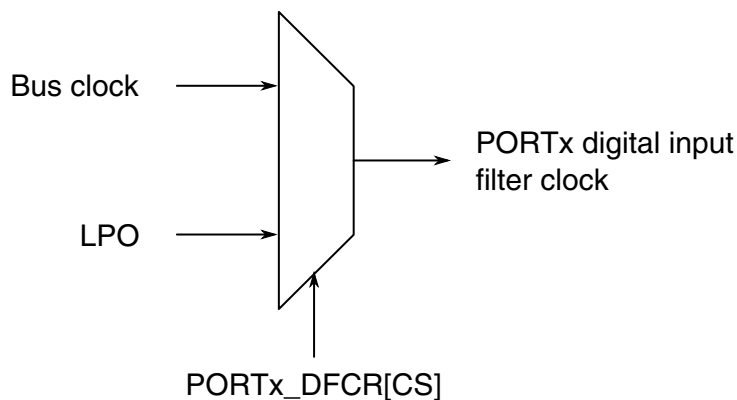


Figure 5-4. PORTx digital input filter clock generation

### 5.7.6 LPTMR clocking

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

**NOTE**

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

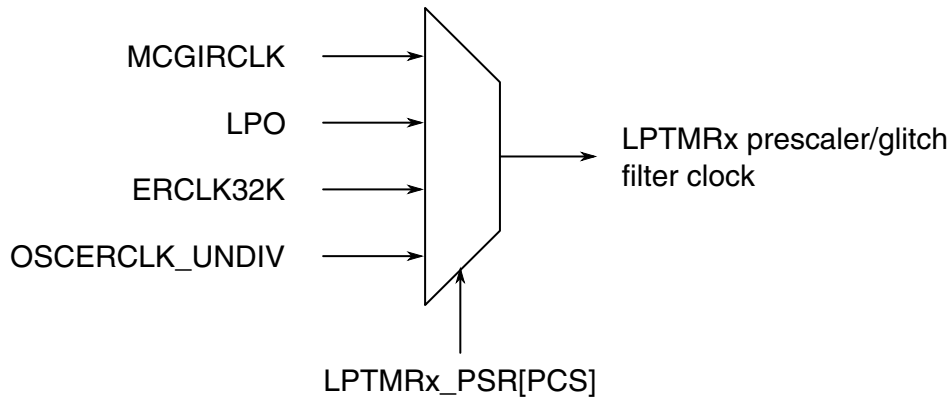


Figure 5-5. LPTMRx prescaler/glitch filter clock generation

### 5.7.7 CLKOUT32K clocking

CLKOUT32K can be driven out on PTE0 as controlled by SIM\_SOPT1[OSC32KOUT], overriding the existing pin mux configuration for that pin. Except for VLLS0 mode, this function is available in all other modes of operation (including LLS3, LLS2, VLLS3, VLLS2, VLLS1 and System Reset).

This function is only available in 64-pin LQFP devices.

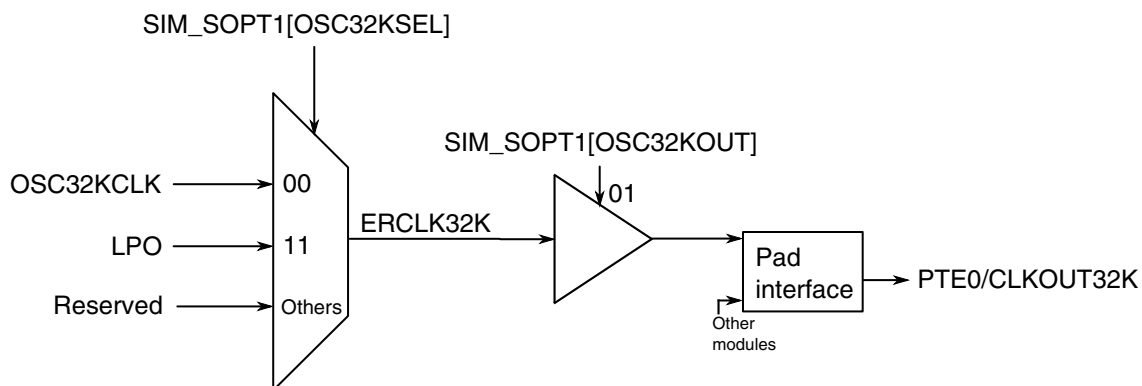


Figure 5-6. CLKOUT32K generation

### 5.7.8 UART clocking

UART0 and UART1 modules operate from the core/system clock, which provides higher performance level for these modules.



# 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	<ul style="list-style-type: none"><li>• Power-on reset (POR)</li></ul>
System resets	<ul style="list-style-type: none"><li>• External pin reset (PIN)</li><li>• Low-voltage detect (LVD)</li><li>• Computer operating properly (COP) watchdog reset</li><li>• Low leakage wakeup (LLWU) reset</li><li>• Multipurpose clock generator loss of clock (LOC) reset</li><li>• Stop mode acknowledge error (SACKERR)</li><li>• Software reset (SW)</li><li>• Lockup reset (LOCKUP)</li><li>• MDM DAP system reset</li></ul>
Debug reset	<ul style="list-style-type: none"><li>• JTAG reset</li><li>• nTRST reset</li></ul>

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 exits reset in functional mode where the CPU is executing code. See [Boot options](#) 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 SRS0 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 JTAG pins have their associated input pins configured as:

- TDI in pull-up (PU)
- TCK in pull-down (PD)
- TMS in PU

and associated output pin configured as:

- TDO with no pull-down or pull-up

Note that the nTRST signal is initially configured as disabled, however once configured to its JTAG functionality its associated input pin is configured as:

- nTRST in PU

### 6.2.2.1 External pin reset (PIN)

On this device,  $\overline{\text{RESET}}$  is a dedicated pin. This pin is open drain and has an internal pullup device. Asserting  $\overline{\text{RESET}}$  wakes the device from any mode. During a pin reset, the RCM's SRS0[PIN] bit is set.

#### 6.2.2.1.1 $\overline{\text{RESET}}$ pin filter

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

For all stop modes where LPO clock is still active (Stop, VLPS, LLS, VLLS3, VLLS2, 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  $\overline{\text{RESET}}$  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 hsrun, normal run, wait, or stop mode. The LVD system is disabled when entering VLPx, LLS, 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 Computer operating properly (COP) watchdog timer

The computer operating properly (COP) 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 COP watchdog. If this periodic refreshing does not occur, the watchdog issues a system reset. The COP reset causes the RCM's SRS0[WDOG] bit to set.

### 6.2.2.4 Low leakage wakeup (LLWU)

The LLWU module provides the means for a number of external pins, the  $\overline{\text{RESET}}$  pin, 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 LLS mode, only the  $\overline{\text{RESET}}$  pin via the LLWU can generate a system reset.
- 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.

### 6.2.2.5 Multipurpose clock generator loss-of-clock (LOC)

The MCG module supports an external reference clock.

If the C6[CME] bit in the MCG module is set, the clock monitor is enabled. If the external reference falls below  $f_{\text{loc\_low}}$  or  $f_{\text{loc\_high}}$ , as controlled by the C2[RANGE] field in the MCG module, the MCU resets. The RCM's SRS0[LOC] bit is set to indicate this reset source.

#### NOTE

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



### 6.2.2.6 Stop mode acknowledge error (SACKERR)

This reset is generated if the core attempts to enter stop mode, 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.

### 6.2.2.7 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.

### 6.2.2.8 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.

### 6.2.2.9 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 JTAG/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.

## 6.2.3 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 RESET\_b 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 RESET\_b 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.

### 6.2.5 Debug resets

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

#### 6.2.5.1 JTAG reset

The JTAG module generate a system reset when certain IR codes are selected. This functional reset is asserted when EXTEST, HIGHZ and CLAMP instructions are active. The reset source from the JTAG module is released when any other IR code is selected. A JTAG reset causes the RCM's SRS1[JTAG] bit to set.

#### 6.2.5.2 nTRST reset

The nTRST pin causes a reset of the JTAG logic when asserted. Asserting the nTRST pin allows the debugger to gain control of the TAP controller state machine (after exiting LLS or VLLSx) without resetting the state of the debug modules.

The nTRST pin does not cause a system reset.

### 6.2.5.3 Resetting the Debug subsystem

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

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

- SWJ-DP
- AHB-AP
- TPIU
- MDM-AP (MDM control and status registers)

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

- CM4 core (core debug registers: DHCSR, DCRSR, DCRDR, DEMCR)
- FPB
- DWT
- ITM
- NVIC
- Crossbar bus switch<sup>1</sup>
- AHB-AP<sup>1</sup>
- Private peripheral bus<sup>1</sup>

## 6.3 Boot

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

### 6.3.1 Boot sources

This device only supports booting from internal flash. Any secondary boot must go through an initialization sequence in flash.

### 6.3.2 Boot options

The device always exits reset in single chip functional mode with the CPU executing code.

---

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

### 6.3.3 FOPT boot options

The flash option register (FOPT) in the flash memory module 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 user can reprogram the option byte in flash to change the FOPT values that are used for subsequent resets. For more details on programming the option byte, refer to the flash memory chapter.

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

#### NOTE

Reserved bits in the option byte should be left in their default erased state of logic 1. FOPT[7:0] = 0x00 is not a valid configuration. FOPT register is written to 0xFF if the contents of NVM's option byte in the flash configuration field is 0x00.

**Table 6-2. Flash Option Register 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.
		1	Fast Initialization. The Flash has faster recoveries at the expense of higher current during these times.
4-3	Reserved		Reserved for future expansion.
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.
		1	NMI pin/interrupts reset default to enabled.
1	Reserved		Reserved for future expansion.

*Table continues on the next page...*

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

Bit Num	Field	Value	Definition
0	LPBOOT		Control the reset value of OUTDIVx values 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.
		0	Low-power boot: OUTDIVx values in SIM_CLKDIV1 register are auto-configured at reset exit for higher divide values that produce lower power consumption at reset exit. <ul style="list-style-type: none"> <li>Core and system clock divider (OUTDIV1) and bus clock divider (OUTDIV2) are 0x7 (divide by 8)</li> <li>Flash clock divider (OUTDIV4) is 0xF (divide by 16)</li> </ul>
		1	Normal boot: OUTDIVx values in SIM_CLKDIV1 register are auto-configured at reset exit for higher frequency values that produce faster operating frequencies at reset exit. <ul style="list-style-type: none"> <li>Core and system clock divider (OUTDIV1) and bus clock divider (OUTDIV2) are 0x0 (divide by 1)</li> <li>Flash clock divider (OUTDIV4) is 0x1 (divide by 2)</li> </ul>

### 6.3.4 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 Mode Controller reset 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 (Core Clock, 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  $\overline{\text{RESET}}$  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 LPBOOT is 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  $\overline{\text{RESET}}$  pin is released. If  $\overline{\text{RESET}}$  continues to be asserted (an indication of a slow rise time on the  $\overline{\text{RESET}}$  pin or external drive in low), the system continues to be held in reset. Once the  $\overline{\text{RESET}}$  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 FOPT[NMI\_DIS] field in the Flash Memory module:
  - If the NMI input is high or the NMI function is disabled in the NMI\_DIS field, the CPU begins execution at the PC location.
  - If the NMI input is low and the NMI function is enabled in the NMI\_DIS field, 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

Information found here 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 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 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 wake-up 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.

## **7.2.2 DMA Wakeup**

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

When a DMA wake-up is detected in PSTOP, STOP or VLPS then the device will initiate 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 will remain in the low power mode with the CPU clock disabled.

During Compute Operation, a DMA wake-up will initiate 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.

Since the DMA wakeup will enable the clocks and negate 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 wake-up 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.

Once the DMA request that initiated the wake-up negates and the DMA completes the current transfer, the device will transition back to 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 would then also enter their appropriate modes.

**NOTE**

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

An enabled DMA wake-up can cause an aborted entry into the low power mode, if the DMA request asserts during the stop mode entry sequence (or reentry if the request asserts during a DMA wakeup) and can cause the SMC to assert its Stop Abort flag. Once the DMA wake-up completes, entry into the low power mode will restart.

An interrupt that occurs during a DMA wake-up will cause an immediate exit from the low power mode (this is optional for Compute Operation) without impacting the DMA transfer.

A DMA wake-up 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, although in general 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 Run mode, HSRUN 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 bus (PPB) remains accessible during Compute Operation, including the MCM, System Control Space (SCS) (for NVIC and FPU), and SysTick. Although access to the GPIO registers 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 memory 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 memory 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 wake-up 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 Description

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 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. The following table compares the various power modes available.

For Run and VLPR 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.

Stop mode entry is not supported directly from HSRUN and requires transition to Run prior to an attempt to enter a stop mode.

## Power Modes Description

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	Default mode out of reset; on-chip voltage regulator is on.	Run	-
High Speed run	Allows maximum performance of chip. In this state, the MCU is able to operate at a faster frequency compared to normal run mode.	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. Reduced frequency Flash access mode (1 MHz); LVD off; internal oscillator provides a low power 4 MHz source for the core, the bus and the peripheral clocks.	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 LPTimer, CMP, DAC can be used. 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
LLS3 (Low Leakage Stop3)	State retention power mode. Most peripherals are in state retention mode (with clocks stopped), but LLWU, LPTimer, CMP, DAC can be used. NVIC is disabled; LLWU is used to wake up.  <b>NOTE:</b> The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit stop mode on an LLS recovery. All SRAM is operating (content retained and I/O states held).	Sleep Deep	Wakeup Interrupt <sup>1</sup>
LLS2 (Low Leakage Stop2)	State retention power mode. Most peripherals are in state retention mode (with clocks stopped), but LLWU, LPTimer, CMP, DAC can be used. NVIC is disabled; LLWU is used to wake up.  <b>NOTE:</b> The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit stop mode on an LLS recovery. A portion of SRAM_U remains powered on (content retained and I/O states held).	Sleep Deep	Wakeup Interrupt <sup>2</sup>

Table continues on the next page...

**Table 7-1. Chip power modes (continued)**

Chip mode	Description	Core mode	Normal recovery method
VLLS3 (Very Low Leakage Stop3)	Most peripherals are disabled (with clocks stopped), but LLWU, LPTimer, CMP, DAC 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 <sup>3</sup>
VLLS2 (Very Low Leakage Stop2)	Most peripherals are disabled (with clocks stopped), but LLWU, LPTimer, CMP, DAC can be used. NVIC is disabled; LLWU is used to wake up.  SRAM_L is powered off. A portion of SRAM_U remains powered on (content retained and I/O states held).	Sleep Deep	Wakeup Reset <sup>3</sup>
VLLS1 (Very Low Leakage Stop1)	Most peripherals are disabled (with clocks stopped), but LLWU, LPTimer, CMP, DAC 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 <sup>3</sup>
VLLS0 (Very Low Leakage Stop 0)	Most peripherals are disabled (with clocks stopped), but LLWU can be used. NVIC is disabled; LLWU is used to wake up.  All of SRAM_U and SRAM_L are powered off.  The POR detect circuit can be optionally powered off.	Sleep Deep	Wakeup Reset <sup>3</sup>

1. Resumes normal run mode operation by executing the LLWU interrupt service routine.
2. Resumes normal run mode operation by executing the LLWU interrupt service routine.
3. 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. The [Nested Vectored Interrupt Controller \(NVIC\)](#) describes interrupt operation and what peripherals can cause interrupts.

### 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.

Recovery from VLLSx is through the wake-up Reset event. The chip wake-ups from VLLSx by means of reset, an enabled pin or enabled module. See the table "LLWU inputs" in the LLWU configuration section for a list of the sources.

The wake-up flow from VLLSx is through reset. The wakeup bit in the SRS registers in the RCM is set indicating that the chip is recovering from a low power mode. Code execution begins; however, the I/O pins are held in their pre low power mode entry

states, and the system oscillator and MCG registers are reset (even if EREFSTEN had been set before entering VLLSx). Software must clear this hold by writing a 1 to the ACKISO bit in the Regulator Status and Control Register in the PMC module.

### NOTE

To avoid unwanted transitions on the pins, software must re-initialize the I/O pins to their pre-low-power mode entry states *before* releasing the hold.

If the oscillator was configured to continue running during VLLSx modes, it must be re-configured before the ACKISO bit is cleared. The oscillator configuration within the MCG is cleared after VLLSx recovery and the oscillator will stop when ACKISO is cleared unless the register is re-configured.

## 7.5 Power mode transitions

The following figure shows the power mode transitions. Any reset always brings the chip back to the normal run state. In run, wait, and stop modes active power regulation is enabled. The VLPx modes offer a lower power operating mode than normal modes. VLPR and VLPW are limited in frequency. The LLS and VLLSx mode(s) are the lowest power stop modes based on amount of logic or memory that is required to be retained by the application.



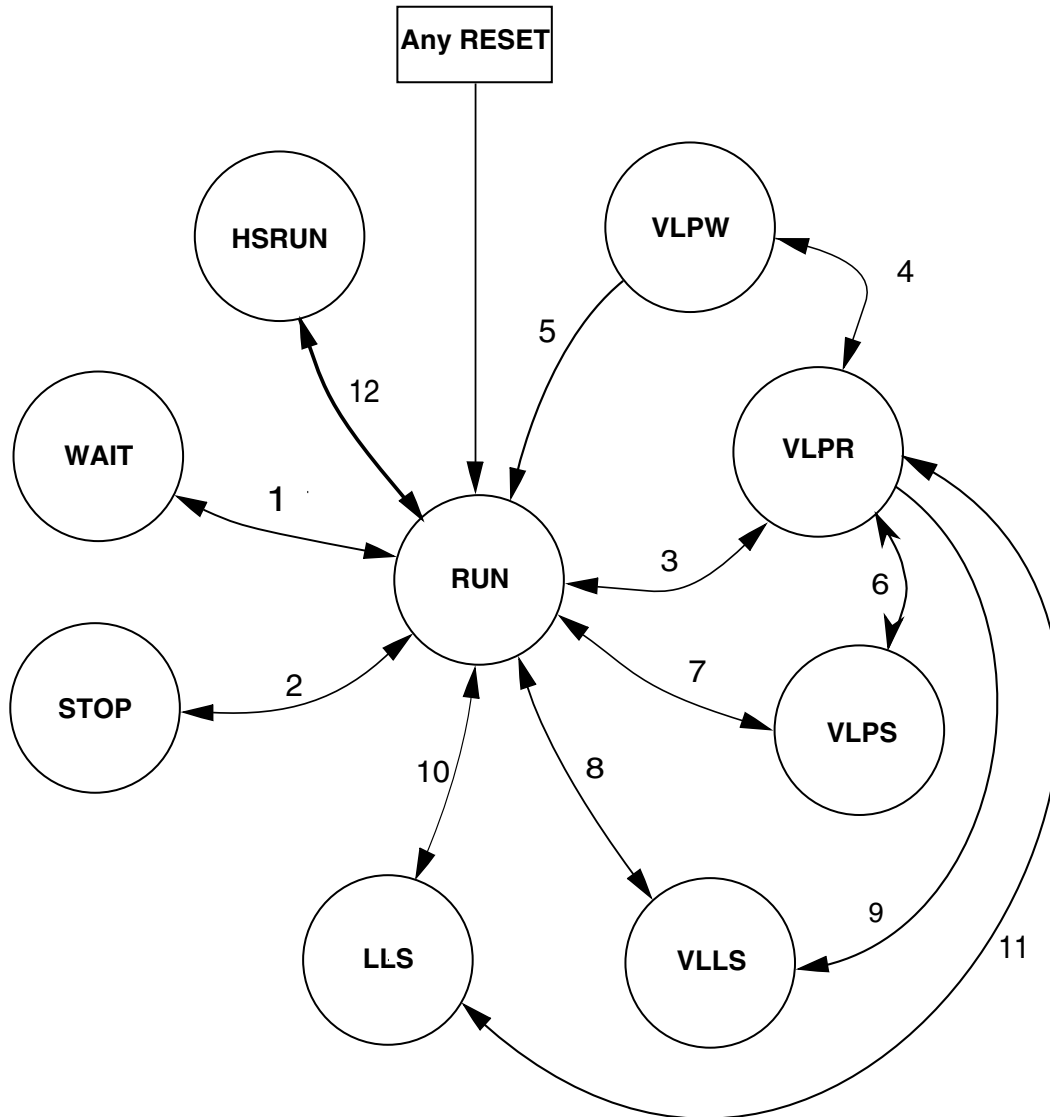


Figure 7-1. Power mode state transition diagram

## 7.6 Power modes shutdown sequencing

When entering stop or other low-power modes, the clocks are shut off in an orderly sequence to safely place the chip in the targeted low-power state. All low-power entry sequences are initiated by the core executing an WFI instruction. The ARM core's outputs, SLEEPDEEP and SLEEPING, trigger entry to the various low-power modes:

- System level wait and VLPW modes equate to: SLEEPING &  $\overline{\text{SLEEPDEEP}}$
- All other low power modes equate to: SLEEPING & SLEEPDEEP

When entering the non-wait modes, the chip performs the following sequence:

- Shuts off Core Clock and System Clock to the ARM Cortex-M4 core immediately.
- Polls stop acknowledge indications from the non-core crossbar masters (DMA), supporting peripherals (SPI, PIT) and the Flash Controller for indications that System Clocks, Bus Clock and/or Flash Clock need to be left enabled to complete a previously initiated operation, effectively stalling entry to the targeted low power mode. When all acknowledges are detected, System Clock, Bus Clock and Flash Clock are turned off at the same time.
- MCG and Mode Controller shut off clock sources and/or the internal supplies driven from the on-chip regulator as defined for the targeted low power mode.

In wait modes, most of the system clocks are not affected by the low power mode entry. The Core Clock to the ARM Cortex-M4 core is shut off. Some modules support stop-in-wait functionality and have their clocks disabled under these configurations.

The debugger modules support a transition from stop, wait, VLPS, and VLPW back to a halted state when the debugger is enabled. This transition is initiated by setting the Debug Request bit in MDM-AP control register. As part of this transition, system clocking is re-established and is equivalent to normal run/VLPR mode clocking configuration.

## 7.7 Flash Program Restrictions

The flash memory on this device should not be programmed or erased while operating in High Speed Run or VLPR power modes.

## 7.8 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 2 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	LLSx	VLLSx
<b>Core modules</b>						
NVIC	FF	FF	static	static	static	OFF
<b>System modules</b>						
Mode Controller	FF	FF	FF	FF	FF	FF
LLWU <sup>1</sup>	static	static	static	static	FF	FF <sup>2</sup>
Regulator	low power	low power	ON	low power	low power	low power in VLLS2/3, OFF in VLLS0/1
LVD	disabled	disabled	ON	disabled	disabled	disabled
Brown-out Detection	ON	ON	ON	ON	ON	ON in VLLS1/2/3, optionally disabled in VLLS0 <sup>3</sup>
DMA	FF Async operation in CPO	FF	Async operation	Async operation	static	OFF
Watchdog	FF	FF	FF	FF	static	OFF
EWM	FF static in CPO	static	static FF in PSTOP2	static	static	OFF
<b>Clocks</b>						
1kHz LPO	ON	ON	ON	ON	ON	ON in VLLS1/2/3, OFF in VLLS0
System oscillator (OSC)	OSCERCLK max of 16 MHz crystal	OSCERCLK max of 16 MHz crystal	OSCERCLK optional	OSCERCLK max of 16 MHz crystal	limited to low range/low power	limited to low range/low power in VLLS1/2/3, OFF in VLLS0
MCG	4 MHz IRC	4 MHz IRC	static - MCGIRCLK optional	static - MCGIRCLK optional (4 MHz IRC only).	static - no clock output	OFF
Core clock	4 MHz max	OFF	OFF	OFF	OFF	OFF
Platform clock	4 MHz max	4 MHz max	OFF	OFF	OFF	OFF
System clock	4 MHz max OFF in CPO	4 MHz max	OFF	OFF	OFF	OFF

Table continues on the next page...

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

Modules	VLPR	VLPW	Stop	VLPS	LLSx	VLLSx
Bus clock	4 MHz max OFF in CPO	4 MHz max	OFF  50 MHz max in PSTOP2 from RUN  4 MHz max in PSTOP2 from VLPR	OFF	OFF	OFF
<b>Memory and memory interfaces</b>						
Flash	1 MHz max access - no program/erase  No register access in CPO	low power	low power	low power	OFF	OFF
System RAM (SRAM_U and SRAM_L) <sup>4</sup>	low power	low power	low power	low power	low power in LLS3, partial in LLS2	low power in VLLS3, partial in VLLS2; otherwise OFF
<b>Communication interfaces</b>						
UART0, UART1	250 kbit/s  static, wakeup on edge in CPO	250 kbit/s	static, wakeup on edge	static, wakeup on edge	static	OFF
SPI	1 Mbit/s (slave)  2 Mbit/s (master)  static in CPO	1 Mbit/s (slave)  2 Mbit/s (master)	static  FF in PSTOP2	static	static	OFF
I <sup>2</sup> C	200 kbit/s  static, address match wakeup in CPO	200 kbit/s	static, address match wakeup  FF in PSTOP2	static, address match wakeup	static	OFF
<b>Security</b>						
CRC	FF  static in CPO	FF	static	static	static	OFF
<b>Timers</b>						
FTM	FF  static in CPO	FF	static  FF in PSTOP2	static	static	OFF
PIT	FF  static in CPO	FF	static  FF in PSTOP2	static	static	OFF
PDB	FF  static in CPO	FF	static  FF in PSTOP2	static	static	OFF
LPTMR	FF	FF	Async operation  FF in PSTOP2	Async operation	Async operation	Async operation <sup>5</sup>
<b>Analog</b>						

Table continues on the next page...

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

Modules	VLPR	VLPW	Stop	VLPS	LLSx	VLLSx
16-bit ADC	FF ADACK and ALTCLK clocks only in CPO	FF	ADACK, ALTCLK, and ALTCLK2 clocks only FF in PSTOP2	ADACK and ALTCLK clocks only	static	OFF
CMP <sup>6</sup>	FF HS or LS compare in CPO	FF	HS or LS compare FF in PSTOP2	HS or LS compare	LS compare	LS compare in VLLS1/2/3, OFF in VLLS0
6-bit DAC	FF static in CPO	FF	static FF in PSTOP2	static	static	static, OFF in VLLS0
VREF	FF	FF	FF	FF	static	OFF
12-bit DAC	FF static in CPO	FF	static FF in PSTOP2	static	static	static
<b>Human-machine interfaces</b>						
GPIO	FF GPIO write only in CPO	FF	static output, wakeup input FF in PSTOP2	static output, wakeup input	static, pins latched	OFF, pins latched

- 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.
- Since LPO clock source is disabled, filters will be bypassed during VLLS0
- The SMC\_STOPCTRL[PORPO] bit in the SMC module controls this option.
- A 8 KB portion of SRAM\_U block is in low power when MCU is in low power modes LLS2 and VLLS2. The remaining System RAM is OFF in LLS2 and VLLS2.
- System OSC and LPO clock sources are not available in VLLS0. Pulse counting is available in all modes.
- CMP in stop or VLPS supports high speed or low speed external pin to pin or external pin to DAC compares. CMP in LLSx or 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, LLSx, 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 via debug. CPU accesses to the flash are not affected by the status of FSEC.

In the unsecured state all flash commands are available to the programming interfaces (JTAG), as well as user code execution of Flash Controller commands. When the flash is secured (FSEC[SEC] = 00, 01, or 11), programmer interfaces are only allowed to launch mass erase operations and have no access to memory locations.

Further information regarding the flash security options and enabling/disabling flash security is available in the [Flash Memory Module](#).

## 8.3 Security Interactions with other Modules

The flash security settings are used by the SoC 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 JTAG port cannot access the memory resources of the MCU. Boundary scan chain operations work, but debugging capabilities are disabled so that the debug port cannot read flash contents.

Although most debug functions are disabled, the debugger can write to the Flash Mass Erase in Progress bit in the MDM-AP Control register 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

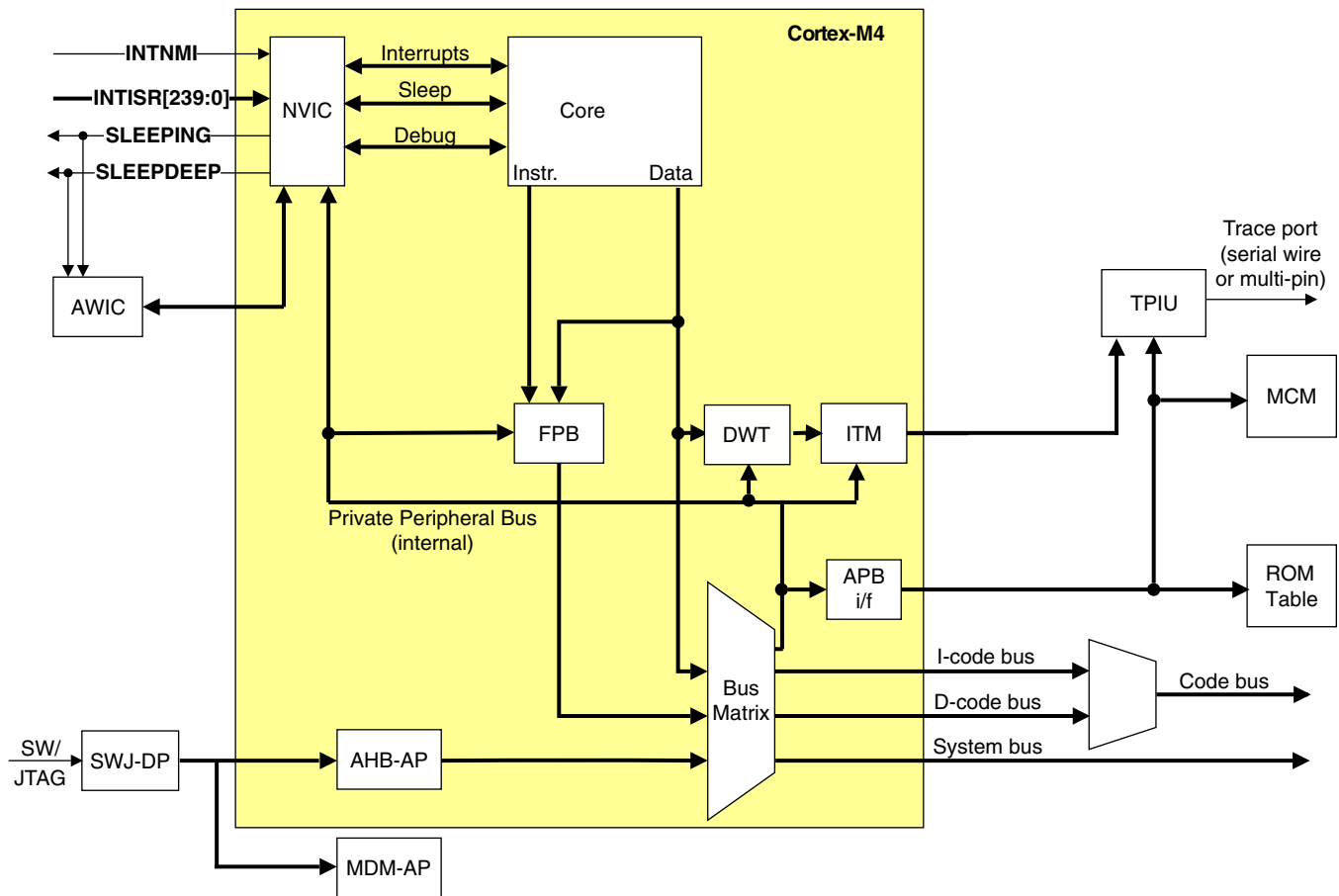
### 9.1 Introduction

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

Four debug interfaces are supported:

- IEEE 1149.1 JTAG
- IEEE 1149.7 JTAG (cJTAG)
- Serial Wire Debug (SWD)
- ARM Real-Time Trace Interface(1-pin asynchronous mode only)

The basic Cortex-M4 debug architecture is very flexible. The following diagram shows the topology of the core debug architecture and its components.



**Figure 9-1. Cortex-M4 Debug Topology**

The following table presents a brief description of each one of the debug components.

**Table 9-1. Debug Components Description**

Module	Description
SWJ-DP+ cJTAG	Modified Debug Port with support for SWD, JTAG, cJTAG
AHB-AP	AHB Master Interface from JTAG to debug module and SOC system memory maps
MDM-AP	Provides centralized control and status registers for an external debugger to control the device.
ROM Table	Identifies which debug IP is available.
Core Debug	Singlestep, Register Access, Run, Core Status
ITM	S/W Instrumentation Messaging + Simple Data Trace Messaging + Watchpoint Messaging
DWT (Data and Address Watchpoints)	4 data and address watchpoints

*Table continues on the next page...*

**Table 9-1. Debug Components Description (continued)**

Module	Description
FPB (Flash Patch and Breakpoints)	<p>The FPB implements hardware breakpoints and patches code and data from code space to system space.</p> <p>The FPB unit contains two literal comparators for matching against literal loads from Code space, and remapping to a corresponding area in System space.</p> <p>The FPB also contains six instruction comparators for matching against instruction fetches from Code space, and remapping to a corresponding area in System space. Alternatively, the six instruction comparators can individually configure the comparators to return a Breakpoint Instruction (BKPT) to the processor core on a match, so providing hardware breakpoint capability.</p>
TPIU (Trace Port Interface Unit)	Asynchronous Mode (1-pin) = TRACE_SWO (available on JTAG_TDO)

## 9.1.1 References

For more information on ARM debug components, see these documents:

- ARMv7-M Architecture Reference Manual
- ARM Debug Interface v5.1
- ARM CoreSight Architecture Specification

## 9.2 The Debug Port

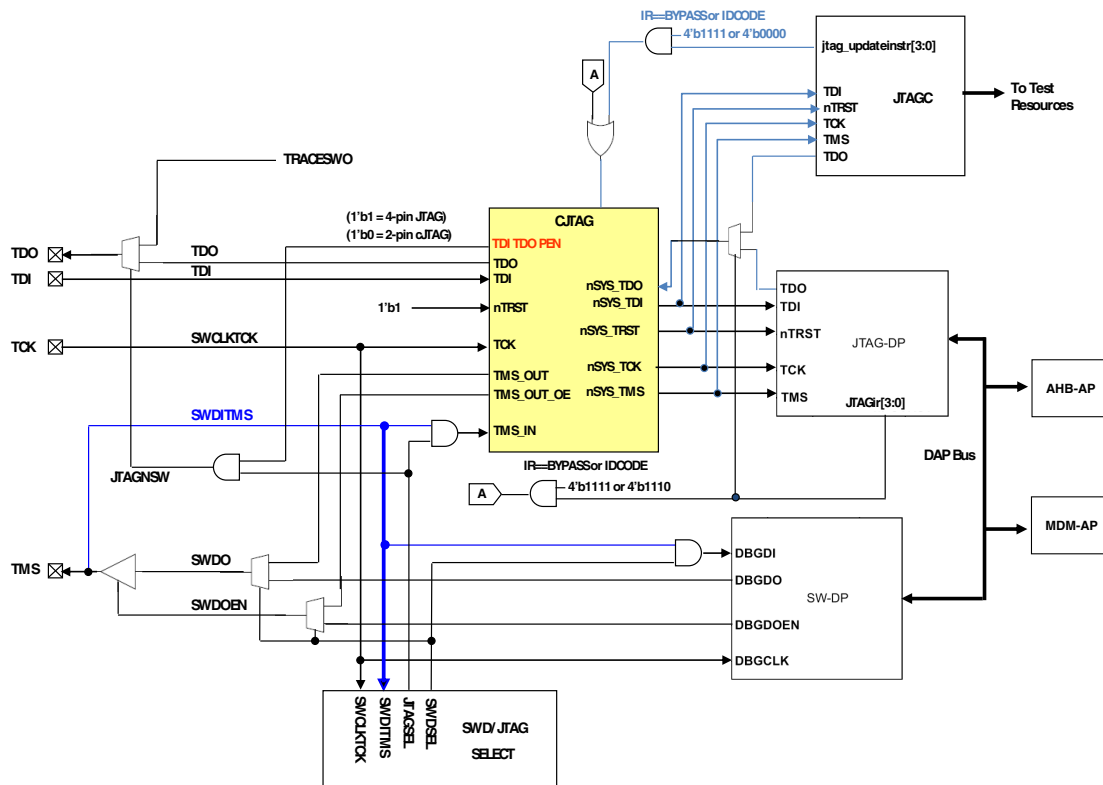


Figure 9-2. Modified Debug Port

The debug port comes out of reset in standard JTAG mode and is switched into either cJTAG or SWD mode by the following sequences. Once the mode has been changed, unused debug pins can be reassigned to any of their alternative muxed functions.

### 9.2.1 JTAG-to-SWD change sequence

1. Send more than 50 TCK cycles with TMS (SWDIO) = 1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111\_1001\_1110\_0111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) = 1

#### NOTE

See the ARM documentation for the CoreSight DAP Lite for restrictions.

## 9.2.2 JTAG-to-cJTAG change sequence

1. Reset the debug port
2. Set the control level to 2 via zero-bit scans
3. Execute the Store Format (STFMT) command (00011) to set the scan format register to 1149.7 scan format

## 9.3 Debug Port Pin Descriptions

The debug port pins default after POR to their JTAG functionality with the exception of JTAG\_TRST\_b and can be later reassigned to their alternate functionalities. In cJTAG and SWD modes JTAG\_TDI and JTAG\_TRST\_b can be configured to alternate GPIO functions.

**Table 9-2. Debug port pins**

Pin Name	JTAG Debug Port		cJTAG Debug Port		SWD Debug Port		Internal Pull-up/Down
	Type	Description	Type	Description	Type	Description	
JTAG_TMS/ SWD_DIO	I/O	JTAG Test Mode Selection	I/O	cJTAG Data	I/O	Serial Wire Data	Pull-up
JTAG_TCLK/ SWD_CLK	I	JTAG Test Clock	I	cJTAG Clock	I	Serial Wire Clock	Pull-down
JTAG_TDI	I	JTAG Test Data Input	-	-	-	-	Pull-up
JTAG_TDO/ TRACE_SWO	O	JTAG Test Data Output	O	Trace output over a single pin	O	Trace output over a single pin	N/C
JTAG_TRST_b	I	JTAG Reset	I	cJTAG Reset	-	-	Pull-up

## 9.4 System TAP connection

The system JTAG controller is connected in parallel to the ARM TAP controller. The system JTAG controller IR codes overlay the ARM JTAG controller IR codes without conflict. Refer to the IR codes table for a list of the available IR codes. The output of the TAPs (TDO) are muxed based on the IR code which is selected. This design is fully JTAG compliant and appears to the JTAG chain as a single TAP. At power on reset, ARM's IDCODE (IR=4'b1110) is selected.

## 9.4.1 IR Codes

**Table 9-3. JTAG Instructions**

Instruction	Code[3:0]	Instruction Summary
IDCODE	0000	Selects device identification register for shift
SAMPLE/PRELOAD	0010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	0011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	0100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
HIGHZ	1001	Selects bypass register while three-stating all output pins and asserting functional reset
CLAMP	1100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ARM_IDCODE	1110	ARM JTAG-DP Instruction
BYPASS	1111	Selects bypass register for data operations
Factory debug reserved	0101, 0110, 0111, 1101	Intended for factory debug only
ARM JTAG-DP Reserved	1000, 1010, 1011, 1110	These instructions will go the ARM JTAG-DP controller. Please look at ARM JTAG-DP documentation for more information on these instructions.
Reserved <sup>1</sup>	All other opcodes	Decoded to select bypass register

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

## 9.5 JTAG 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 get 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 (DAP) using JTAG, cJTAG, or SWD. The MDM-AP is accessible as Debug Access Port 1 with the available registers shown in the table below.

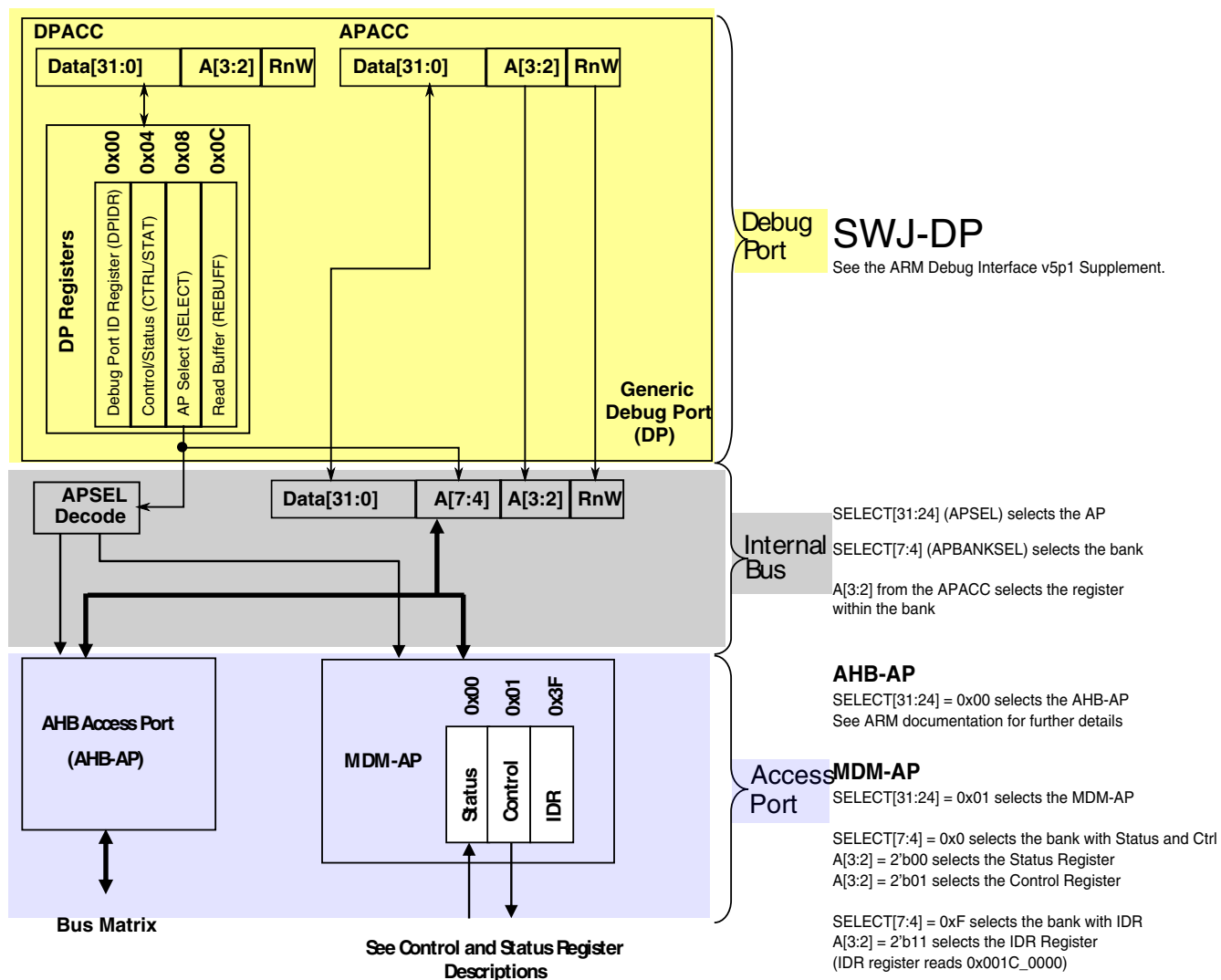
**Table 9-4. MDM-AP Register Summary**

Address	Register	Description
---------	----------	-------------

*Table continues on the next page...*

**Table 9-4. MDM-AP Register Summary (continued)**

0x0100_0000	Status	See <a href="#">MDM-AP Status Register</a>
0x0100_0004	Control	See <a href="#">MDM-AP Control Register</a>
0x0100_00FC	ID	Read-only identification register that always reads as 0x001C_0000



**Figure 9-3. MDM AP Addressing**

## 9.5.1 MDM-AP Control Register

**Table 9-5. MDM-AP Control register assignments**

Bit	Name	Secure <sup>1</sup>	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 and SEC settings), 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.  If the Core is in a stop or wait mode, this bit can be used to wakeup the core and transition to a halted state.
3	System Reset Request	N	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. Once 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 holds the in reset when VLLSx modes are exited to allow the debugger time to re-initialize debug IP before the debug session continues.  The Mode Controller captures this bit logic on entry to VLLSx modes. Upon exit from VLLSx modes, the Mode Controller will hold the 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 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 re-initializes all debug IP and then assert this control bit to allow the Mode Controller to release the 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	LLS, VLLSx Status Acknowledge	N	Set this bit to acknowledge the DAP LLS and VLLS Status bits have been read. This acknowledge automatically clears the status bits.  This bit is used by the debugger to clear the sticky LLS and VLLSx mode entry status bits. This bit is asserted and cleared by the debugger.

*Table continues on the next page...*



**Table 9-5. MDM-AP Control register assignments (continued)**

Bit	Name	Secure <sup>1</sup>	Description
8	Timestamp Disable	N	Set this bit to disable the 48-bit global trace timestamp counter during debug halt mode when the core is halted.  0 The timestamp counter continues to count assuming trace is enabled. (default)  1 The timestamp counter freezes when the core has halted (debug halt mode).
9 – 31	Reserved for future use	N	

1. Command available in secure mode

## 9.5.2 MDM-AP Status Register

**Table 9-6. MDM-AP Status register assignments**

Bit	Name	Description
0	Flash Mass Erase Acknowledge	The Flash Mass Erase Acknowledge bit is cleared after any system reset. The bit is also cleared at launch of a mass erase command due to write of Flash Mass Erase in Progress bit in 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 and SEC settings), an erase request due to setting of Flash Mass Erase in Progress bit is not acknowledged.
1	Flash Ready	Indicate Flash has been initialized and debugger can be configured even if system is continuing to be held in reset via 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 if the MCU can be mass erased or not  0 Mass erase is disabled  1 Mass erase is enabled
6	Backdoor Access Key Enable	Indicates if the MCU has the backdoor access key enabled.  0 Disabled  1 Enabled

*Table continues on the next page...*

Table 9-6. MDM-AP Status register assignments (continued)

Bit	Name	Description
7	LP Enabled	Decode of LPLLSM control bits to indicate that VLPS, LLS, 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 conjunction 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 or not.  This bit is used to throttle JTAG TCK frequency up/down.
9	LLS Mode Exit	This bit indicates an exit from LLS mode has occurred. The debugger will lose communication while the system is in LLS (including access to this register). Once communication is reestablished, this bit indicates that the system had been in LLS. Since the debug modules held their state during LLS, they do not need to be reconfigured.  This bit is set during the LLS recovery sequence. The LLS Mode Exit bit is held until the debugger has had a chance to recognize that LLS was exited and is cleared by a write of 1 to the LLS, VLLSx Status Acknowledge bit in MDM AP Control register.
10	VLLSx Modes Exit	This bit indicates an exit from VLLSx mode has occurred. The debugger will lose communication while the system is in VLLSx (including access to this register). Once communication is reestablished, this bit indicates that the system had been in VLLSx. Since the debug modules lose their state during VLLSx modes, they need to be reconfigured.  This bit is set during the VLLSx recovery sequence. The VLLSx Mode Exit bit is held until the debugger has had a chance to recognize that a VLLS mode was exited and is cleared by a write of 1 to the LLS, VLLSx Status Acknowledge bit in 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.

## 9.6 Debug Resets

The debug system receives the following sources of reset:

- JTAG\_TRST\_b from an external signal. This signal is optional and may not be available in all packages.

- Debug reset (CDBGSTREQ bit within the SWJ-DP CTRL/STAT register) in the TCLK domain that allows the debugger to reset the debug logic.
- TRST asserted via the cJTAG escape command.
- System POR reset

Conversely the debug system is capable of generating system reset using the following mechanism:

- 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.

## 9.7 AHB-AP

AHB-AP provides the debugger access to all memory and registers in the system, including processor registers through the NVIC. System access is independent of the processor status. AHB-AP does not do back-to-back transactions on the bus, so all transactions are non-sequential. AHB-AP can perform unaligned and bit-band transactions. AHB-AP transactions bypass the FPB, so the FPB cannot remap AHB-AP transactions. SWJ/SW-DP-initiated transaction aborts drive an AHB-AP-supported sideband signal called HABORT. This signal is driven into the Bus Matrix, which resets the Bus Matrix state, so that AHB-AP can access the Private Peripheral Bus for last ditch debugging such as read/stop/reset the core. AHB-AP transactions are little endian.

For a short period at the start of a system reset event the system security status is being determined and debugger access to all AHB-AP transactions is blocked. The MDM-AP Status register is accessible and can be monitored to determine when this initial period is completed. After this initial period, if system reset is held via assertion of the RESET pin, the debugger has access via the bus matrix to the private peripheral bus to configure the debug IP even while system reset is asserted. While in system reset, access to other memory and register resources, accessed over the Crossbar Switch, is blocked.

## 9.8 ITM

The ITM is an application-driven trace source that supports printf style debugging to trace Operating System (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets. There are four sources that can

generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. Software trace -- Software can write directly to ITM stimulus registers. This emits packets.
2. Hardware trace -- The DWT generates these packets, and the ITM emits them.
3. Time stamping -- Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex-M4 clock or the bitclock rate of the Serial Wire Viewer (SWV) output clocks the counter.
4. Global system timestamping. Timestamps can optionally be generated using a system-wide 48-bit count value.

## 9.9 Core Trace Connectivity

The ITM can route its data to the TPIU. (See the [MCM \(Miscellaneous Control Module\)](#) for controlling the routing to the TPIU.) This configuration enables the use of trace with low cost tools while maintaining the compatibility with trace probes.

## 9.10 TPIU

The TPIU acts as a bridge between the on-chip trace data from the Instrumentation Trace Macrocell (ITM) to a data stream, encapsulating IDs where required, that is then captured by a Trace Port Analyzer (TPA). The TPIU is specially designed for low-cost debug.

## 9.11 DWT

The DWT is a unit that performs the following debug functionality:

- It contains four comparators that you can configure as a hardware watchpoint, a PC sampler event trigger, or a data address sampler event trigger. The first comparator, DWT\_COMP0, can also compare against the clock cycle counter, CYCCNT. The second comparator, DWT\_COMP1, can also be used as a data comparator.
- The DWT contains counters for:
  - Clock cycles (CYCCNT)
  - Folded instructions
  - Load store unit (LSU) operations
  - Sleep cycles

- CPI (all instruction cycles except for the first cycle)
- Interrupt overhead

### NOTE

An event is emitted each time a counter overflows.

- The DWT can be configured to emit PC samples at defined intervals, and to emit interrupt event information.

## 9.12 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 once 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 are not allowed and 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 but also result in the system entering Stop mode instead.

In VLLS mode all debug modules are powered off and reset at wakeup. In LLS mode, the debug modules retain their state but no debug activity is possible.

### NOTE

When using cJTAG and entering LLS mode, the cJTAG controller must be reset on exit from LLS mode.

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 HW, the MDM-AP Control register can be configured hold the system in reset on recovery so that the debugger can regain control and reconfigure debug logic prior to the system exiting reset and resuming operation.

## 9.12.1 Debug Module State in Low Power Modes

The following table shows the state of the debug modules in low power modes. 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.
- static = Module register states and associated memories are retained.
- OFF = Modules are powered off; module is in reset state upon wakeup.

**Table 9-7. Debug Module State in Low Power Modes**

Module	STOP	VLPR	VLPW	VLPS	LLS	VLLSx
Debug Port	FF	FF	FF	OFF	static	OFF
AHB-AP	FF	FF	FF	OFF	static	OFF
ITM	FF	FF	FF	OFF	static	OFF
TPIU	FF	FF	FF	OFF	static	OFF
DWT	FF	FF	FF	OFF	static	OFF

## 9.13 Debug & Security

When 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, the debugger also has the capability of performing a mass erase operation via writes to the MDM-AP Control Register. 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.

# Chapter 10

## Signal Multiplexing and Signal Descriptions

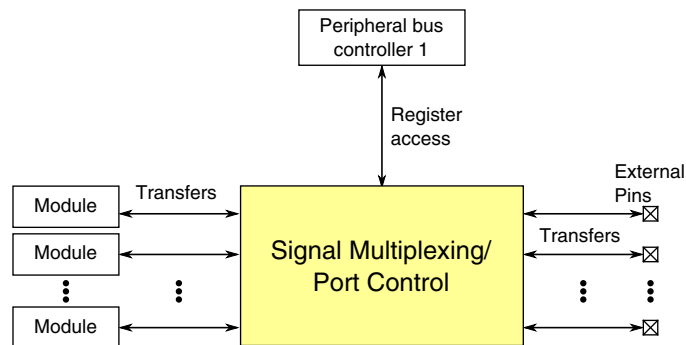
### 10.1 Introduction

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	<a href="#">Port control</a>
System memory map		<a href="#">System memory map</a>

*Table continues on the next page...*

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

Topic	Related module	Reference
Clocking		<a href="#">Clock Distribution</a>
Register access	Peripheral bus controller	<a href="#">Peripheral bridge</a>

## 10.2.1 Port control and interrupt module features

- 32-pin ports

### NOTE

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

- Each 32-pin port is assigned one interrupt.

**Table 10-2. Ports summary**

Feature	Port A	Port B	Port C	Port D	Port E
Pull select control	Yes	Yes	Yes	Yes	Yes
Pull select at reset	PTA1/PTA2/PTA3/ PTA4/PTA5=Pull up, Others=Pull down	Pull down	Pull down	Pull down	Pull down
Pull enable control	Yes	Yes	Yes	Yes	Yes
Pull enable at reset	PTA0/PTA1/PTA2/ PTA3/ PTA4=Enabled; Others=Disabled	Disabled	Disabled	Disabled	Disabled
Slew rate enable control	Yes	Yes	Yes	Yes	Yes
Slew rate enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Passive filter enable control	PTA4=Yes; Others=No	No	No	No	No
Passive filter enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Open drain enable control	Yes	Yes	Yes	Yes	Yes
Open drain enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Drive strength enable control	No	PTB0/PTB1 only	PTC3/PTC4 only	PTD4/PTD5/PTD6/ PTD7 only	No
Drive strength enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled

*Table continues on the next page...*



**Table 10-2. Ports summary (continued)**

Feature	Port A	Port B	Port C	Port D	Port E
Pin mux control	Yes	Yes	Yes	Yes	Yes
Pin mux at reset	PTA0/PTA1/PTA2/ PTA3/PTA4=ALT7; Others=ALT0	ALT0	ALT0	ALT0	ALT0
Lock bit	Yes	Yes	Yes	Yes	Yes
Interrupt and DMA request	Yes	Yes	Yes	Yes	Yes
Digital glitch filter	No	No	No	Yes	No

## 10.2.2 Clock gating

The clock to the port control module can be gated on and off using the SCGC5[PORTx] bits in the SIM module. These bits are cleared after any reset, which disables the clock to the corresponding module to conserve power. Prior to initializing the corresponding module, set SCGC5[PORTx] in the SIM module 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 K02F 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

The 48-pin LQFP package offering is subject to removal.

## Pinout

64 LQFP	48 LQFP	32 QFN	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
1	—	—	PTE0/ CLKOUT32K	DISABLED		PTE0/ CLKOUT32K		UART1_TX				
2	—	—	PTE1/ LLWU_P0	DISABLED		PTE1/ LLWU_P0		UART1_RX				
3	1	1	VDD	VDD	VDD							
4	2	2	VSS	VSS	VSS							
5	3	3	PTE16	ADC0_SE4a/ ADC0_DP1	ADC0_SE4a/ ADC0_DP1	PTE16	SPI0_PCS0	UART1_TX	FTM_CLKIN0		FTM0_FLT3	
6	4	4	PTE17	ADC0_SE5a/ ADC0_DM1	ADC0_SE5a/ ADC0_DM1	PTE17	SPI0_SCK	UART1_RX	FTM_CLKIN1		LPTMR0_ALT3	
7	5	5	PTE18	ADC0_SE6a/ ADC0_DP2	ADC0_SE6a/ ADC0_DP2	PTE18	SPI0_SOUT	UART1_CTS_b	I2C0_SDA			
8	6	6	PTE19	ADC0_SE7a/ ADC0_DM2	ADC0_SE7a/ ADC0_DM2	PTE19	SPI0_SIN	UART1_RTS_b	I2C0_SCL			
9	7	—	ADC0_DP0	ADC0_DP0	ADC0_DP0							
10	8	—	ADC0_DM0	ADC0_DM0	ADC0_DM0							
11	—	—	ADC0_DP3	ADC0_DP3	ADC0_DP3							
12	—	—	ADC0_DM3	ADC0_DM3	ADC0_DM3							
13	9	7	VDDA	VDDA	VDDA							
14	10	7	VREFH	VREFH	VREFH							
15	11	8	VREFL	VREFL	VREFL							
16	12	8	VSSA	VSSA	VSSA							
17	13	—	VREF_OUT/ CMP1_IN5/ CMP0_IN5	VREF_OUT/ CMP1_IN5/ CMP0_IN5	VREF_OUT/ CMP1_IN5/ CMP0_IN5							
18	14	9	DAC0_OUT/ CMP1_IN3/ ADC0_SE23	DAC0_OUT/ CMP1_IN3/ ADC0_SE23	DAC0_OUT/ CMP1_IN3/ ADC0_SE23							
19	—	—	CMP0_IN4	CMP0_IN4	CMP0_IN4							
20	15	10	PTE24	ADC0_SE17	ADC0_SE17	PTE24			I2C0_SCL		EWM_OUT_b	
21	16	11	PTE25	ADC0_SE18	ADC0_SE18	PTE25			I2C0_SDA		EWM_IN	
22	17	12	PTA0	JTAG_TCLK/ SWD_CLK		PTA0	UART0_CTS_b	FTM0_CH5				JTAG_TCLK/ SWD_CLK
23	18	13	PTA1	JTAG_TDI		PTA1	UART0_RX					JTAG_TDI
24	19	14	PTA2	JTAG_TDO/ TRACE_SWO		PTA2	UART0_TX					JTAG_TDO/ TRACE_SWO
25	20	15	PTA3	JTAG_TMS/ SWD_DIO		PTA3	UART0_RTS_b	FTM0_CH0				JTAG_TMS/ SWD_DIO
26	21	16	PTA4/ LLWU_P3	NMI_b		PTA4/ LLWU_P3		FTM0_CH1				NMI_b
27	—	—	PTA5	DISABLED		PTA5		FTM0_CH2				JTAG_TRST_b
28	—	—	PTA12	DISABLED		PTA12		FTM1_CH0				FTM1_QD_ PHA
29	—	—	PTA13/ LLWU_P4	DISABLED		PTA13/ LLWU_P4		FTM1_CH1				FTM1_QD_ PHB
30	22	—	VDD	VDD	VDD							

## Chapter 10 Signal Multiplexing and Signal Descriptions

64 LQFP	48 LQFP	32 QFN	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
31	23	—	VSS	VSS	VSS							
32	24	17	PTA18	EXTAL0	EXTAL0	PTA18		FTM0_FLT2	FTM_CLKIN0			
33	25	18	PTA19	XTAL0	XTAL0	PTA19		FTM1_FLT0	FTM_CLKIN1		LPTMR0_ALT1	
34	26	19	RESET_b	RESET_b	RESET_b							
35	27	20	PTB0/ LLWU_P5	ADC0_SE8	ADC0_SE8	PTB0/ LLWU_P5	I2C0_SCL	FTM1_CH0			FTM1_QD_ PHA	
36	28	21	PTB1	ADC0_SE9	ADC0_SE9	PTB1	I2C0_SDA	FTM1_CH1			FTM1_QD_ PHB	
37	29	—	PTB2	ADC0_SE12	ADC0_SE12	PTB2	I2C0_SCL	UART0_RTS_b			FTM0_FLT3	
38	30	—	PTB3	ADC0_SE13	ADC0_SE13	PTB3	I2C0_SDA	UART0_CTS_b			FTM0_FLT0	
39	31	—	PTB16	DISABLED		PTB16		UART0_RX	FTM_CLKIN0		EWM_IN	
40	32	—	PTB17	DISABLED		PTB17		UART0_TX	FTM_CLKIN1		EWM_OUT_b	
41	—	—	PTB18	DISABLED		PTB18		FTM2_CH0			FTM2_QD_ PHA	
42	—	—	PTB19	DISABLED		PTB19		FTM2_CH1			FTM2_QD_ PHB	
43	33	—	PTC0	ADC0_SE14	ADC0_SE14	PTC0	SPI0_PCS4	PDB0_EXTRG				
44	34	22	PTC1/ LLWU_P6	ADC0_SE15	ADC0_SE15	PTC1/ LLWU_P6	SPI0_PCS3	UART1_RTS_b	FTM0_CH0			
45	35	23	PTC2	ADC0_SE4b/ CMP1_IN0	ADC0_SE4b/ CMP1_IN0	PTC2	SPI0_PCS2	UART1_CTS_b	FTM0_CH1			
46	36	24	PTC3/ LLWU_P7	CMP1_IN1	CMP1_IN1	PTC3/ LLWU_P7	SPI0_PCS1	UART1_RX	FTM0_CH2	CLKOUT		
47	—	—	VSS	VSS	VSS							
48	—	—	VDD	VDD	VDD							
49	37	25	PTC4/ LLWU_P8	DISABLED		PTC4/ LLWU_P8	SPI0_PCS0	UART1_TX	FTM0_CH3		CMP1_OUT	
50	38	26	PTC5/ LLWU_P9	DISABLED		PTC5/ LLWU_P9	SPI0_SCK	LPTMR0_ALT2			CMP0_OUT	FTM0_CH2
51	39	27	PTC6/ LLWU_P10	CMP0_IN0	CMP0_IN0	PTC6/ LLWU_P10	SPI0_SOUT	PDB0_EXTRG				
52	40	28	PTC7	CMP0_IN1	CMP0_IN1	PTC7	SPI0_SIN					
53	—	—	PTC8	CMP0_IN2	CMP0_IN2	PTC8						
54	—	—	PTC9	CMP0_IN3	CMP0_IN3	PTC9					FTM2_FLT0	
55	—	—	PTC10	DISABLED		PTC10						
56	—	—	PTC11/ LLWU_P11	DISABLED		PTC11/ LLWU_P11						
57	41	—	PTD0/ LLWU_P12	DISABLED		PTD0/ LLWU_P12	SPI0_PCS0					
58	42	—	PTD1	ADC0_SE5b	ADC0_SE5b	PTD1	SPI0_SCK					
59	43	—	PTD2/ LLWU_P13	DISABLED		PTD2/ LLWU_P13	SPI0_SOUT					I2C0_SCL
60	44	—	PTD3	DISABLED		PTD3	SPI0_SIN					I2C0_SDA
61	45	29	PTD4/ LLWU_P14	DISABLED		PTD4/ LLWU_P14	SPI0_PCS1	UART0_RTS_b	FTM0_CH4		EWM_IN	

## Pinout

64 LQFP	48 LQFP	32 QFN	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
62	46	30	PTD5	ADC0_SE6b	ADC0_SE6b	PTD5	SPI0_PCS2	UART0_CTS_b	FTM0_CH5		EWM_OUT_b	
63	47	31	PTD6/ LLWU_P15	ADC0_SE7b	ADC0_SE7b	PTD6/ LLWU_P15	SPI0_PCS3	UART0_RX	FTM0_CH0		FTM0_FLT0	
64	48	32	PTD7	DISABLED		PTD7		UART0_TX	FTM0_CH1		FTM0_FLT1	

### 10.3.2 K02F Pinouts

The below 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.

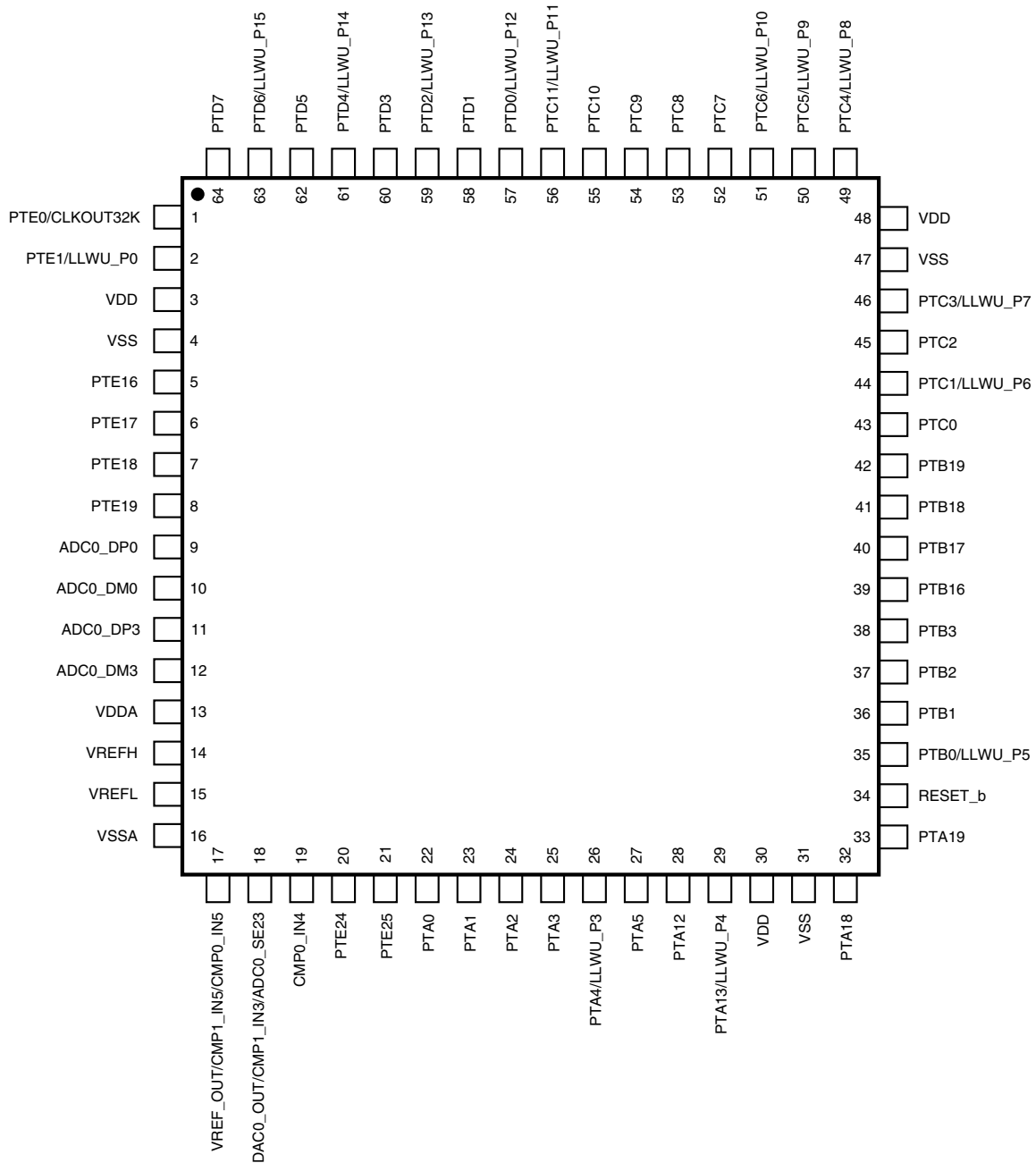
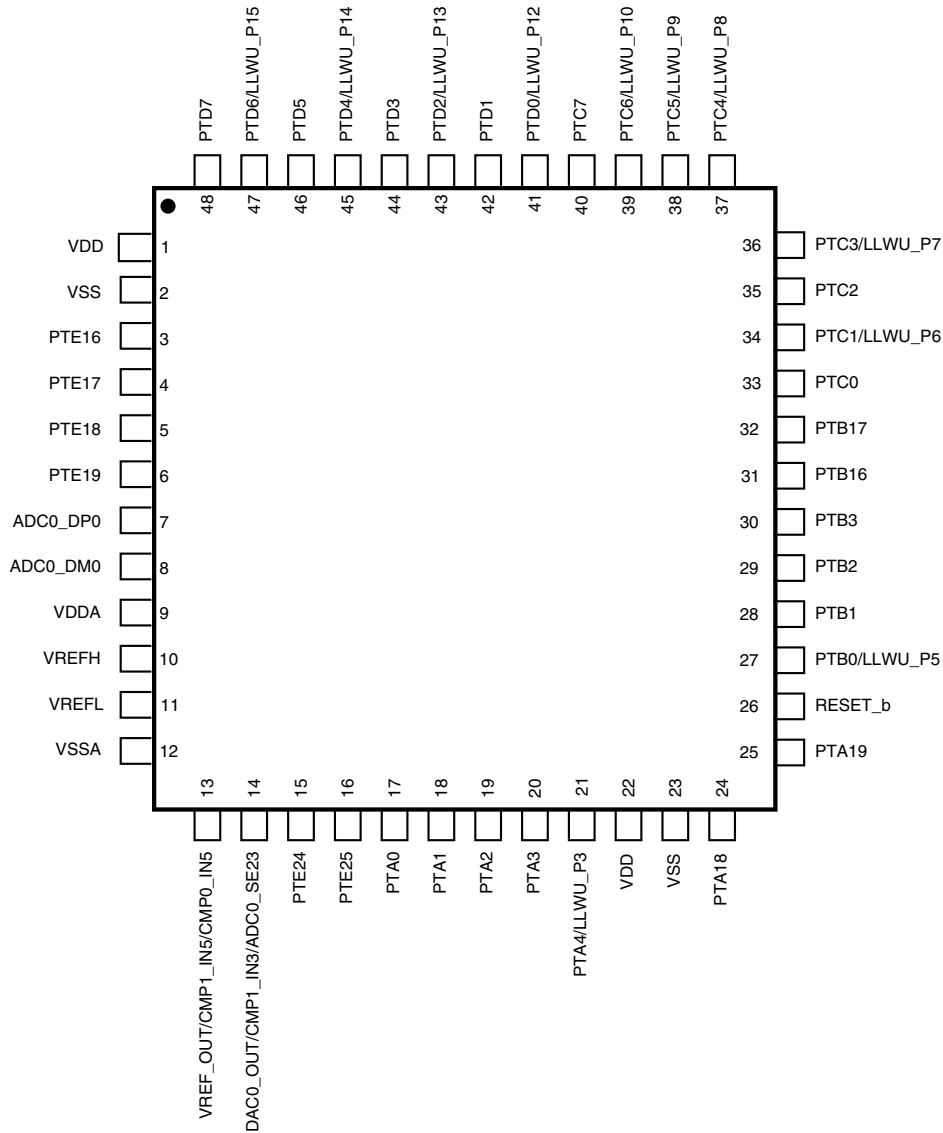


Figure 10-2. K02F 64 LQFP Pinout Diagram



**Figure 10-3. K02F 48 LQFP Pinout Diagram**

**NOTE**

The 48-pin LQFP package offering is subject to removal.

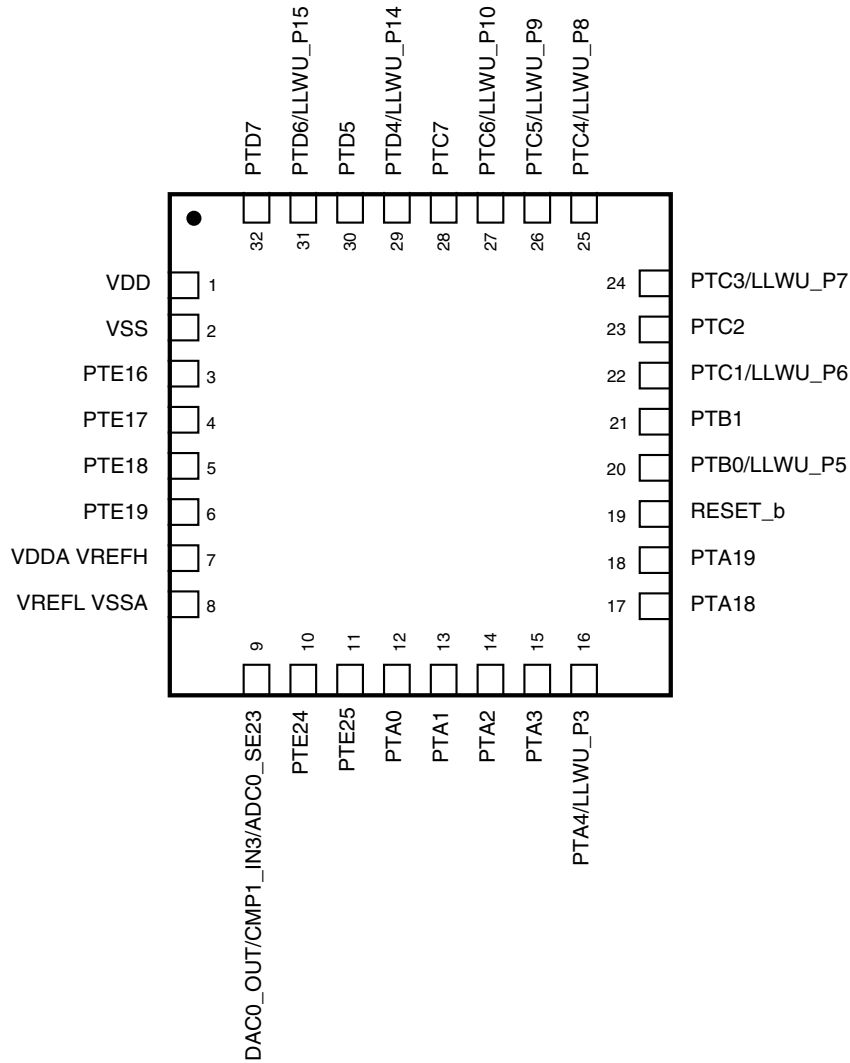


Figure 10-4. K02F 32 QFN Pinout Diagram

## 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. JTAG Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
JTAG_TMS	JTAG_TMS/ SWD_DIO	JTAG Test Mode Selection	I/O
JTAG_TCLK	JTAG_TCLK/ SWD_CLK	JTAG Test Clock	I
JTAG_TDI	JTAG_TDI	JTAG Test Data Input	I
JTAG_TDO	JTAG_TDO/ TRACE_SWO	JTAG Test Data Output	O
JTAG_TRST	JTAG_TRST_b	JTAG Reset	I

**Table 10-4. SWD Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
SWD_DIO	JTAG_TMS/ SWD_DIO	Serial Wire Data	I/O
SWD_CLK	JTAG_TCLK/ SWD_CLK	Serial Wire Clock	I

**Table 10-5. TPIU Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
TRACE_SWO	JTAG_TDO/ TRACE_SWO	Trace output data from the ARM CoreSight debug block over a single pin	O

## 10.4.2 System Modules

**Table 10-6. EWM Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
EWM_IN	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
EWM_OUT	EWM_out	EWM reset out signal	O



## 10.4.3 Clock Modules

**Table 10-7. OSC Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
EXTALO	EXTAL	External clock/Oscillator input	I
XTALO	XTAL	Oscillator output	O

## 10.4.4 Analog

**Table 10-8. ADC 0 Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
ADC0_DP[3:0]	DADP3–DADP0	Differential Analog Channel Inputs	I
ADC0_DM[3:0]	DADM3–DADM0	Differential Analog Channel Inputs	I
ADC0_SE <sub>n</sub>	AD <sub>n</sub>	Single-Ended Analog Channel Inputs	I
VREFH	V <sub>REFSH</sub>	Voltage Reference Select High	I
VREFL	V <sub>REFSL</sub>	Voltage Reference Select Low	I
VDDA	V <sub>DDA</sub>	Analog Power Supply	I
VSSA	V <sub>SSA</sub>	Analog Ground	I

**Table 10-9. CMP 0 Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
CMP0_IN[5:0]	IN[5:0]	Analog voltage inputs	I
CMP0_OUT	CMPO	Comparator output	O

**Table 10-10. CMP 1 Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
CMP1_IN[5:0]	IN[5:0]	Analog voltage inputs	I
CMP1_OUT	CMPO	Comparator output	O

**Table 10-11. DAC 0 Signal Descriptions**

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

**Table 10-12. VREF Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
VREF_OUT	VREF_OUT	Internally-generated Voltage Reference output	O

## 10.4.5 Timer Modules

**Table 10-13. FTM 0 Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
FTM_CLKIN[1:0]	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 10-14. FTM 1 Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
FTM_CLKIN[1:0]	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 7-0	I/O
FTM1_FLT0	FAULTj	Fault input (j), where j can be 3-0	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 10-15. FTM 2 Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
FTM_CLKIN[1:0]	EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I
FTM2_CH[1:0]	CHn	FTM channel (n), where n can be 7-0	I/O
FTM2_FLT0	FAULTj	Fault input (j), where j can be 3-0	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 10-16. PDB 0 Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
PDB0_EXTRG	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

**Table 10-17. LPTMR 0 Signal Descriptions**

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

## 10.4.6 Communication Interfaces

**Table 10-18. 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_PCS5	PCS5/ $\overline{PCSS}$	Peripheral Chip Select 5 /Peripheral Chip Select Strobe	O
SPI0_SIN	SIN	Serial Data In	I
SPI0_SOUT	SOUT	Serial Data Out	O
SPI0_SCK	SCK	Serial Clock (O)	I/O

**Table 10-19. I<sup>2</sup>C 0 Signal Descriptions**

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

**Table 10-20. UART 0 Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
UART0_CTS	CTS	Clear to send	I
UART0_RTS	RTS	Request to send	O
UART0_TX	TXD	Transmit data	O
UART0_RX	RXD	Receive data	I

**Table 10-21. UART 1 Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
UART1_CTS	CTS	Clear to send	I
UART1_RTS	RTS	Request to send	O
UART1_TX	TXD	Transmit data	O
UART1_RX	RXD	Receive data	I

## 10.4.7 Human-Machine Interfaces (HMI)

**Table 10-22. GPIO Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
PTA[31:0] <sup>1</sup>	PORTA31–PORTA0	General-purpose input/output	I/O
PTB[31:0] <sup>1</sup>	PORTB31–PORTB0	General-purpose input/output	I/O
PTC[31:0] <sup>1</sup>	PORTC31–PORTC0	General-purpose input/output	I/O
PTD[31:0] <sup>1</sup>	PORTD31–PORTD0	General-purpose input/output	I/O
PTE[31:0] <sup>1</sup>	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, digital filtering, 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
  - 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
- Digital input filter on selected pins

- Digital input filter for each pin, usable by any digital peripheral muxed onto the pin
- Individual enable or bypass control field per pin
- Selectable clock source for digital input filter with a five bit resolution on filter size
- Functional in all digital pin multiplexing modes
- Port control
  - Individual pull control fields with pullup, pulldown, and pull-disable support
  - Individual drive strength field supporting high and low drive strength
  - Individual slew rate field supporting fast and slow slew rates
  - Individual input passive filter field supporting enable and disable of the individual input passive filter
  - Individual open drain field supporting enable and disable of the individual open drain output
  - 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.

In Stop mode, the digital input filters are bypassed unless they are configured to run from the 1-kHz LPO clock source.

### 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	11.5.1/198
4004_9004	Pin Control Register n (PORTA_PCR1)	32	R/W	See section	11.5.1/198
4004_9008	Pin Control Register n (PORTA_PCR2)	32	R/W	See section	11.5.1/198
4004_900C	Pin Control Register n (PORTA_PCR3)	32	R/W	See section	11.5.1/198
4004_9010	Pin Control Register n (PORTA_PCR4)	32	R/W	See section	11.5.1/198
4004_9014	Pin Control Register n (PORTA_PCR5)	32	R/W	See section	11.5.1/198
4004_9018	Pin Control Register n (PORTA_PCR6)	32	R/W	See section	11.5.1/198
4004_901C	Pin Control Register n (PORTA_PCR7)	32	R/W	See section	11.5.1/198
4004_9020	Pin Control Register n (PORTA_PCR8)	32	R/W	See section	11.5.1/198
4004_9024	Pin Control Register n (PORTA_PCR9)	32	R/W	See section	11.5.1/198
4004_9028	Pin Control Register n (PORTA_PCR10)	32	R/W	See section	11.5.1/198
4004_902C	Pin Control Register n (PORTA_PCR11)	32	R/W	See section	11.5.1/198
4004_9030	Pin Control Register n (PORTA_PCR12)	32	R/W	See section	11.5.1/198
4004_9034	Pin Control Register n (PORTA_PCR13)	32	R/W	See section	11.5.1/198
4004_9038	Pin Control Register n (PORTA_PCR14)	32	R/W	See section	11.5.1/198
4004_903C	Pin Control Register n (PORTA_PCR15)	32	R/W	See section	11.5.1/198
4004_9040	Pin Control Register n (PORTA_PCR16)	32	R/W	See section	11.5.1/198
4004_9044	Pin Control Register n (PORTA_PCR17)	32	R/W	See section	11.5.1/198
4004_9048	Pin Control Register n (PORTA_PCR18)	32	R/W	See section	11.5.1/198
4004_904C	Pin Control Register n (PORTA_PCR19)	32	R/W	See section	11.5.1/198
4004_9050	Pin Control Register n (PORTA_PCR20)	32	R/W	See section	11.5.1/198
4004_9054	Pin Control Register n (PORTA_PCR21)	32	R/W	See section	11.5.1/198
4004_9058	Pin Control Register n (PORTA_PCR22)	32	R/W	See section	11.5.1/198
4004_905C	Pin Control Register n (PORTA_PCR23)	32	R/W	See section	11.5.1/198
4004_9060	Pin Control Register n (PORTA_PCR24)	32	R/W	See section	11.5.1/198
4004_9064	Pin Control Register n (PORTA_PCR25)	32	R/W	See section	11.5.1/198
4004_9068	Pin Control Register n (PORTA_PCR26)	32	R/W	See section	11.5.1/198
4004_906C	Pin Control Register n (PORTA_PCR27)	32	R/W	See section	11.5.1/198
4004_9070	Pin Control Register n (PORTA_PCR28)	32	R/W	See section	11.5.1/198
4004_9074	Pin Control Register n (PORTA_PCR29)	32	R/W	See section	11.5.1/198
4004_9078	Pin Control Register n (PORTA_PCR30)	32	R/W	See section	11.5.1/198
4004_907C	Pin Control Register n (PORTA_PCR31)	32	R/W	See section	11.5.1/198
4004_9080	Global Pin Control Low Register (PORTA_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/200
4004_9084	Global Pin Control High Register (PORTA_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/201
4004_90A0	Interrupt Status Flag Register (PORTA_ISFR)	32	w1c	0000_0000h	11.5.4/202

Table continues on the next page...



## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_90C0	Digital Filter Enable Register (PORTA_DFER)	32	R/W	0000_0000h	11.5.5/202
4004_90C4	Digital Filter Clock Register (PORTA_DFCL)	32	R/W	0000_0000h	11.5.6/203
4004_90C8	Digital Filter Width Register (PORTA_DFWR)	32	R/W	0000_0000h	11.5.7/203
4004_A000	Pin Control Register n (PORTB_PCR0)	32	R/W	See section	11.5.1/198
4004_A004	Pin Control Register n (PORTB_PCR1)	32	R/W	See section	11.5.1/198
4004_A008	Pin Control Register n (PORTB_PCR2)	32	R/W	See section	11.5.1/198
4004_A00C	Pin Control Register n (PORTB_PCR3)	32	R/W	See section	11.5.1/198
4004_A010	Pin Control Register n (PORTB_PCR4)	32	R/W	See section	11.5.1/198
4004_A014	Pin Control Register n (PORTB_PCR5)	32	R/W	See section	11.5.1/198
4004_A018	Pin Control Register n (PORTB_PCR6)	32	R/W	See section	11.5.1/198
4004_A01C	Pin Control Register n (PORTB_PCR7)	32	R/W	See section	11.5.1/198
4004_A020	Pin Control Register n (PORTB_PCR8)	32	R/W	See section	11.5.1/198
4004_A024	Pin Control Register n (PORTB_PCR9)	32	R/W	See section	11.5.1/198
4004_A028	Pin Control Register n (PORTB_PCR10)	32	R/W	See section	11.5.1/198
4004_A02C	Pin Control Register n (PORTB_PCR11)	32	R/W	See section	11.5.1/198
4004_A030	Pin Control Register n (PORTB_PCR12)	32	R/W	See section	11.5.1/198
4004_A034	Pin Control Register n (PORTB_PCR13)	32	R/W	See section	11.5.1/198
4004_A038	Pin Control Register n (PORTB_PCR14)	32	R/W	See section	11.5.1/198
4004_A03C	Pin Control Register n (PORTB_PCR15)	32	R/W	See section	11.5.1/198
4004_A040	Pin Control Register n (PORTB_PCR16)	32	R/W	See section	11.5.1/198
4004_A044	Pin Control Register n (PORTB_PCR17)	32	R/W	See section	11.5.1/198
4004_A048	Pin Control Register n (PORTB_PCR18)	32	R/W	See section	11.5.1/198
4004_A04C	Pin Control Register n (PORTB_PCR19)	32	R/W	See section	11.5.1/198
4004_A050	Pin Control Register n (PORTB_PCR20)	32	R/W	See section	11.5.1/198
4004_A054	Pin Control Register n (PORTB_PCR21)	32	R/W	See section	11.5.1/198
4004_A058	Pin Control Register n (PORTB_PCR22)	32	R/W	See section	11.5.1/198
4004_A05C	Pin Control Register n (PORTB_PCR23)	32	R/W	See section	11.5.1/198
4004_A060	Pin Control Register n (PORTB_PCR24)	32	R/W	See section	11.5.1/198
4004_A064	Pin Control Register n (PORTB_PCR25)	32	R/W	See section	11.5.1/198
4004_A068	Pin Control Register n (PORTB_PCR26)	32	R/W	See section	11.5.1/198
4004_A06C	Pin Control Register n (PORTB_PCR27)	32	R/W	See section	11.5.1/198
4004_A070	Pin Control Register n (PORTB_PCR28)	32	R/W	See section	11.5.1/198
4004_A074	Pin Control Register n (PORTB_PCR29)	32	R/W	See section	11.5.1/198
4004_A078	Pin Control Register n (PORTB_PCR30)	32	R/W	See section	11.5.1/198
4004_A07C	Pin Control Register n (PORTB_PCR31)	32	R/W	See section	11.5.1/198
4004_A080	Global Pin Control Low Register (PORTB_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/200

Table continues on the next page...

## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_A084	Global Pin Control High Register (PORTB_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/201
4004_A0A0	Interrupt Status Flag Register (PORTB_ISFR)	32	w1c	0000_0000h	11.5.4/202
4004_A0C0	Digital Filter Enable Register (PORTB_DFER)	32	R/W	0000_0000h	11.5.5/202
4004_A0C4	Digital Filter Clock Register (PORTB_DFCR)	32	R/W	0000_0000h	11.5.6/203
4004_A0C8	Digital Filter Width Register (PORTB_DFWR)	32	R/W	0000_0000h	11.5.7/203
4004_B000	Pin Control Register n (PORTC_PCR0)	32	R/W	See section	11.5.1/198
4004_B004	Pin Control Register n (PORTC_PCR1)	32	R/W	See section	11.5.1/198
4004_B008	Pin Control Register n (PORTC_PCR2)	32	R/W	See section	11.5.1/198
4004_B00C	Pin Control Register n (PORTC_PCR3)	32	R/W	See section	11.5.1/198
4004_B010	Pin Control Register n (PORTC_PCR4)	32	R/W	See section	11.5.1/198
4004_B014	Pin Control Register n (PORTC_PCR5)	32	R/W	See section	11.5.1/198
4004_B018	Pin Control Register n (PORTC_PCR6)	32	R/W	See section	11.5.1/198
4004_B01C	Pin Control Register n (PORTC_PCR7)	32	R/W	See section	11.5.1/198
4004_B020	Pin Control Register n (PORTC_PCR8)	32	R/W	See section	11.5.1/198
4004_B024	Pin Control Register n (PORTC_PCR9)	32	R/W	See section	11.5.1/198
4004_B028	Pin Control Register n (PORTC_PCR10)	32	R/W	See section	11.5.1/198
4004_B02C	Pin Control Register n (PORTC_PCR11)	32	R/W	See section	11.5.1/198
4004_B030	Pin Control Register n (PORTC_PCR12)	32	R/W	See section	11.5.1/198
4004_B034	Pin Control Register n (PORTC_PCR13)	32	R/W	See section	11.5.1/198
4004_B038	Pin Control Register n (PORTC_PCR14)	32	R/W	See section	11.5.1/198
4004_B03C	Pin Control Register n (PORTC_PCR15)	32	R/W	See section	11.5.1/198
4004_B040	Pin Control Register n (PORTC_PCR16)	32	R/W	See section	11.5.1/198
4004_B044	Pin Control Register n (PORTC_PCR17)	32	R/W	See section	11.5.1/198
4004_B048	Pin Control Register n (PORTC_PCR18)	32	R/W	See section	11.5.1/198
4004_B04C	Pin Control Register n (PORTC_PCR19)	32	R/W	See section	11.5.1/198
4004_B050	Pin Control Register n (PORTC_PCR20)	32	R/W	See section	11.5.1/198
4004_B054	Pin Control Register n (PORTC_PCR21)	32	R/W	See section	11.5.1/198
4004_B058	Pin Control Register n (PORTC_PCR22)	32	R/W	See section	11.5.1/198
4004_B05C	Pin Control Register n (PORTC_PCR23)	32	R/W	See section	11.5.1/198
4004_B060	Pin Control Register n (PORTC_PCR24)	32	R/W	See section	11.5.1/198
4004_B064	Pin Control Register n (PORTC_PCR25)	32	R/W	See section	11.5.1/198
4004_B068	Pin Control Register n (PORTC_PCR26)	32	R/W	See section	11.5.1/198
4004_B06C	Pin Control Register n (PORTC_PCR27)	32	R/W	See section	11.5.1/198
4004_B070	Pin Control Register n (PORTC_PCR28)	32	R/W	See section	11.5.1/198
4004_B074	Pin Control Register n (PORTC_PCR29)	32	R/W	See section	11.5.1/198
4004_B078	Pin Control Register n (PORTC_PCR30)	32	R/W	See section	11.5.1/198

Table continues on the next page...

## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_B07C	Pin Control Register n (PORTC_PCR31)	32	R/W	See section	11.5.1/198
4004_B080	Global Pin Control Low Register (PORTC_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/200
4004_B084	Global Pin Control High Register (PORTC_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/201
4004_B0A0	Interrupt Status Flag Register (PORTC_ISFR)	32	w1c	0000_0000h	11.5.4/202
4004_B0C0	Digital Filter Enable Register (PORTC_DFER)	32	R/W	0000_0000h	11.5.5/202
4004_B0C4	Digital Filter Clock Register (PORTC_DFCR)	32	R/W	0000_0000h	11.5.6/203
4004_B0C8	Digital Filter Width Register (PORTC_DFWR)	32	R/W	0000_0000h	11.5.7/203
4004_C000	Pin Control Register n (PORTD_PCR0)	32	R/W	See section	11.5.1/198
4004_C004	Pin Control Register n (PORTD_PCR1)	32	R/W	See section	11.5.1/198
4004_C008	Pin Control Register n (PORTD_PCR2)	32	R/W	See section	11.5.1/198
4004_C00C	Pin Control Register n (PORTD_PCR3)	32	R/W	See section	11.5.1/198
4004_C010	Pin Control Register n (PORTD_PCR4)	32	R/W	See section	11.5.1/198
4004_C014	Pin Control Register n (PORTD_PCR5)	32	R/W	See section	11.5.1/198
4004_C018	Pin Control Register n (PORTD_PCR6)	32	R/W	See section	11.5.1/198
4004_C01C	Pin Control Register n (PORTD_PCR7)	32	R/W	See section	11.5.1/198
4004_C020	Pin Control Register n (PORTD_PCR8)	32	R/W	See section	11.5.1/198
4004_C024	Pin Control Register n (PORTD_PCR9)	32	R/W	See section	11.5.1/198
4004_C028	Pin Control Register n (PORTD_PCR10)	32	R/W	See section	11.5.1/198
4004_C02C	Pin Control Register n (PORTD_PCR11)	32	R/W	See section	11.5.1/198
4004_C030	Pin Control Register n (PORTD_PCR12)	32	R/W	See section	11.5.1/198
4004_C034	Pin Control Register n (PORTD_PCR13)	32	R/W	See section	11.5.1/198
4004_C038	Pin Control Register n (PORTD_PCR14)	32	R/W	See section	11.5.1/198
4004_C03C	Pin Control Register n (PORTD_PCR15)	32	R/W	See section	11.5.1/198
4004_C040	Pin Control Register n (PORTD_PCR16)	32	R/W	See section	11.5.1/198
4004_C044	Pin Control Register n (PORTD_PCR17)	32	R/W	See section	11.5.1/198
4004_C048	Pin Control Register n (PORTD_PCR18)	32	R/W	See section	11.5.1/198
4004_C04C	Pin Control Register n (PORTD_PCR19)	32	R/W	See section	11.5.1/198
4004_C050	Pin Control Register n (PORTD_PCR20)	32	R/W	See section	11.5.1/198
4004_C054	Pin Control Register n (PORTD_PCR21)	32	R/W	See section	11.5.1/198
4004_C058	Pin Control Register n (PORTD_PCR22)	32	R/W	See section	11.5.1/198
4004_C05C	Pin Control Register n (PORTD_PCR23)	32	R/W	See section	11.5.1/198
4004_C060	Pin Control Register n (PORTD_PCR24)	32	R/W	See section	11.5.1/198
4004_C064	Pin Control Register n (PORTD_PCR25)	32	R/W	See section	11.5.1/198
4004_C068	Pin Control Register n (PORTD_PCR26)	32	R/W	See section	11.5.1/198
4004_C06C	Pin Control Register n (PORTD_PCR27)	32	R/W	See section	11.5.1/198

Table continues on the next page...

## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_C070	Pin Control Register n (PORTD_PCR28)	32	R/W	See section	11.5.1/198
4004_C074	Pin Control Register n (PORTD_PCR29)	32	R/W	See section	11.5.1/198
4004_C078	Pin Control Register n (PORTD_PCR30)	32	R/W	See section	11.5.1/198
4004_C07C	Pin Control Register n (PORTD_PCR31)	32	R/W	See section	11.5.1/198
4004_C080	Global Pin Control Low Register (PORTD_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/200
4004_C084	Global Pin Control High Register (PORTD_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/201
4004_C0A0	Interrupt Status Flag Register (PORTD_ISFR)	32	w1c	0000_0000h	11.5.4/202
4004_C0C0	Digital Filter Enable Register (PORTD_DFER)	32	R/W	0000_0000h	11.5.5/202
4004_C0C4	Digital Filter Clock Register (PORTD_DFCR)	32	R/W	0000_0000h	11.5.6/203
4004_C0C8	Digital Filter Width Register (PORTD_DFWR)	32	R/W	0000_0000h	11.5.7/203
4004_D000	Pin Control Register n (PORTE_PCR0)	32	R/W	See section	11.5.1/198
4004_D004	Pin Control Register n (PORTE_PCR1)	32	R/W	See section	11.5.1/198
4004_D008	Pin Control Register n (PORTE_PCR2)	32	R/W	See section	11.5.1/198
4004_D00C	Pin Control Register n (PORTE_PCR3)	32	R/W	See section	11.5.1/198
4004_D010	Pin Control Register n (PORTE_PCR4)	32	R/W	See section	11.5.1/198
4004_D014	Pin Control Register n (PORTE_PCR5)	32	R/W	See section	11.5.1/198
4004_D018	Pin Control Register n (PORTE_PCR6)	32	R/W	See section	11.5.1/198
4004_D01C	Pin Control Register n (PORTE_PCR7)	32	R/W	See section	11.5.1/198
4004_D020	Pin Control Register n (PORTE_PCR8)	32	R/W	See section	11.5.1/198
4004_D024	Pin Control Register n (PORTE_PCR9)	32	R/W	See section	11.5.1/198
4004_D028	Pin Control Register n (PORTE_PCR10)	32	R/W	See section	11.5.1/198
4004_D02C	Pin Control Register n (PORTE_PCR11)	32	R/W	See section	11.5.1/198
4004_D030	Pin Control Register n (PORTE_PCR12)	32	R/W	See section	11.5.1/198
4004_D034	Pin Control Register n (PORTE_PCR13)	32	R/W	See section	11.5.1/198
4004_D038	Pin Control Register n (PORTE_PCR14)	32	R/W	See section	11.5.1/198
4004_D03C	Pin Control Register n (PORTE_PCR15)	32	R/W	See section	11.5.1/198
4004_D040	Pin Control Register n (PORTE_PCR16)	32	R/W	See section	11.5.1/198
4004_D044	Pin Control Register n (PORTE_PCR17)	32	R/W	See section	11.5.1/198
4004_D048	Pin Control Register n (PORTE_PCR18)	32	R/W	See section	11.5.1/198
4004_D04C	Pin Control Register n (PORTE_PCR19)	32	R/W	See section	11.5.1/198
4004_D050	Pin Control Register n (PORTE_PCR20)	32	R/W	See section	11.5.1/198
4004_D054	Pin Control Register n (PORTE_PCR21)	32	R/W	See section	11.5.1/198
4004_D058	Pin Control Register n (PORTE_PCR22)	32	R/W	See section	11.5.1/198
4004_D05C	Pin Control Register n (PORTE_PCR23)	32	R/W	See section	11.5.1/198
4004_D060	Pin Control Register n (PORTE_PCR24)	32	R/W	See section	11.5.1/198

Table continues on the next page...

## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_D064	Pin Control Register n (PORTE_PCR25)	32	R/W	See section	11.5.1/198
4004_D068	Pin Control Register n (PORTE_PCR26)	32	R/W	See section	11.5.1/198
4004_D06C	Pin Control Register n (PORTE_PCR27)	32	R/W	See section	11.5.1/198
4004_D070	Pin Control Register n (PORTE_PCR28)	32	R/W	See section	11.5.1/198
4004_D074	Pin Control Register n (PORTE_PCR29)	32	R/W	See section	11.5.1/198
4004_D078	Pin Control Register n (PORTE_PCR30)	32	R/W	See section	11.5.1/198
4004_D07C	Pin Control Register n (PORTE_PCR31)	32	R/W	See section	11.5.1/198
4004_D080	Global Pin Control Low Register (PORTE_GPCLR)	32	W (always reads 0)	0000_0000h	11.5.2/200
4004_D084	Global Pin Control High Register (PORTE_GPCHR)	32	W (always reads 0)	0000_0000h	11.5.3/201
4004_D0A0	Interrupt Status Flag Register (PORTE_ISFR)	32	w1c	0000_0000h	11.5.4/202
4004_D0C0	Digital Filter Enable Register (PORTE_DFER)	32	R/W	0000_0000h	11.5.5/202
4004_D0C4	Digital Filter Clock Register (PORTE_DFCR)	32	R/W	0000_0000h	11.5.6/203
4004_D0C8	Digital Filter Width Register (PORTE_DFWR)	32	R/W	0000_0000h	11.5.7/203

### 11.5.1 Pin Control Register n (PORTx\_PCRn)

#### 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 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 + (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	[shaded]							w1c	[shaded]				IRQC			
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	LK	0				MUX			0	DSE	ODE	PFE	0	SRE	PE	PS
W	LK	[shaded]				MUX			[shaded]	DSE	ODE	PFE	[shaded]	SRE	PE	PS
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\_PCRn 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 The pin interrupt configuration is valid in all digital pin muxing modes.

Table continues on the next page...

**PORTx\_PCRn field descriptions (continued)**

Field	Description
	<p>0 Configured interrupt is not detected.</p> <p>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.</p>
23–20 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
19–16 IRQC	<p>Interrupt Configuration</p> <p>The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt/DMA request as follows:</p> <p>0000 Interrupt/DMA request disabled.</p> <p>0001 DMA request on rising edge.</p> <p>0010 DMA request on falling edge.</p> <p>0011 DMA request on either edge.</p> <p>1000 Interrupt when logic 0.</p> <p>1001 Interrupt on rising-edge.</p> <p>1010 Interrupt on falling-edge.</p> <p>1011 Interrupt on either edge.</p> <p>1100 Interrupt when logic 1.</p> <p>Others Reserved.</p>
15 LK	<p>Lock Register</p> <p>0 Pin Control Register fields [15:0] are not locked.</p> <p>1 Pin Control Register fields [15:0] are locked and cannot be updated until the next system reset.</p>
14–11 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
10–8 MUX	<p>Pin Mux Control</p> <p>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.</p> <p>The corresponding pin is configured in the following pin muxing slot as follows:</p> <p>000 Pin disabled (analog).</p> <p>001 Alternative 1 (GPIO).</p> <p>010 Alternative 2 (chip-specific).</p> <p>011 Alternative 3 (chip-specific).</p> <p>100 Alternative 4 (chip-specific).</p> <p>101 Alternative 5 (chip-specific).</p> <p>110 Alternative 6 (chip-specific).</p> <p>111 Alternative 7 (chip-specific).</p>
7 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
6 DSE	<p>Drive Strength Enable</p> <p>Drive strength configuration is valid in all digital pin muxing modes.</p>

*Table continues on the next page...*

**PORTx\_PCRn field descriptions (continued)**

Field	Description
	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 ODE	Open Drain Enable Open drain configuration is valid in all digital pin muxing modes. 0 Open drain output is disabled on the corresponding pin. 1 Open drain output is enabled on the corresponding pin, if the pin is configured as a digital output.
4 PFE	Passive Filter Enable 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 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 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 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	<p>Global Pin Write Enable</p> <p>Selects which Pin Control Registers (15 through 0) bits [15:0] update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.</p> <p>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.</p>
15–0 GPWD	<p>Global Pin Write Data</p> <p>Write value that is written to all Pin Control Registers bits [15:0] that are selected by GPWE.</p>

**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

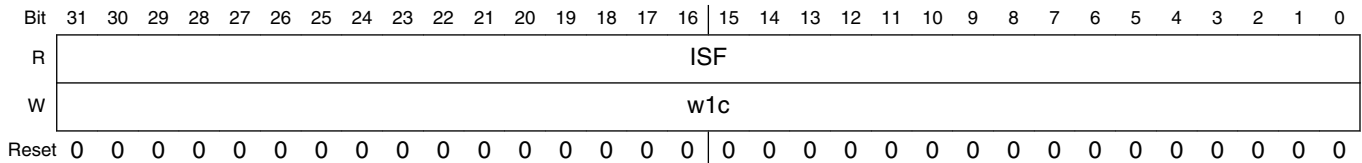
**PORTx\_GPCHR field descriptions**

Field	Description
31–16 GPWE	<p>Global Pin Write Enable</p> <p>Selects which Pin Control Registers (31 through 16) bits [15:0] update with the value in GPWD. If a selected Pin Control Register is locked then the write to that register is ignored.</p> <p>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.</p>
15–0 GPWD	<p>Global Pin Write Data</p> <p>Write value that is written to all Pin Control Registers bits [15:0] that are selected by GPWE.</p>

### 11.5.4 Interrupt Status Flag Register (PORTx\_ISFR)

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



#### PORTx\_ISFR field descriptions

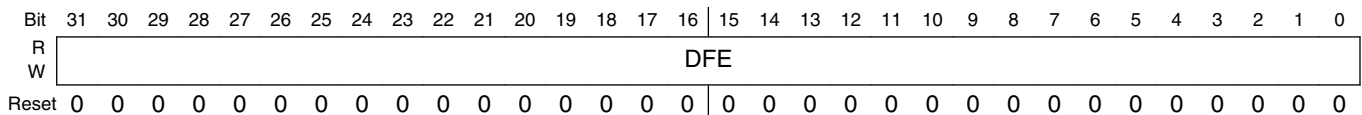
Field	Description
31–0 ISF	<p>Interrupt Status Flag</p> <p>Each bit in the field indicates the detection of the configured interrupt of the same number as the field.</p> <p>0 Configured interrupt is not detected.</p> <p>1 Configured interrupt is detected. If the pin is configured to generate a DMA request, 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.</p>

### 11.5.5 Digital Filter Enable Register (PORTx\_DFER)

The corresponding bit is read only for pins that do not support a digital filter. Refer to the Chapter of Signal Multiplexing and Signal Descriptions for the pins that support digital filter.

The digital filter configuration is valid in all digital pin muxing modes.

Address: Base address + C0h offset



#### PORTx\_DFER field descriptions

Field	Description
31–0 DFE	Digital Filter Enable

**PORTx\_DFER field descriptions (continued)**

Field	Description
	The digital filter configuration is valid in all digital pin muxing modes. The output of each digital filter is reset to zero at system reset and whenever the digital filter is disabled. Each bit in the field enables the digital filter of the same number as the field.
0	Digital filter is disabled on the corresponding pin and output of the digital filter is reset to zero.
1	Digital filter is enabled on the corresponding pin, if the pin is configured as a digital input.

**11.5.6 Digital Filter Clock Register (PORTx\_DFRCR)**

This register is read only for ports that do not support a digital filter.

The digital filter configuration is valid in all digital pin muxing modes.

Address: Base address + C4h 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																
W																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

**PORTx\_DFRCR field descriptions**

Field	Description
31–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 CS	Clock Source The digital filter configuration is valid in all digital pin muxing modes. Configures the clock source for the digital input filters. Changing the filter clock source must be done only when all digital filters are disabled. 0 Digital filters are clocked by the bus clock. 1 Digital filters are clocked by the 1-kHz LPO clock.

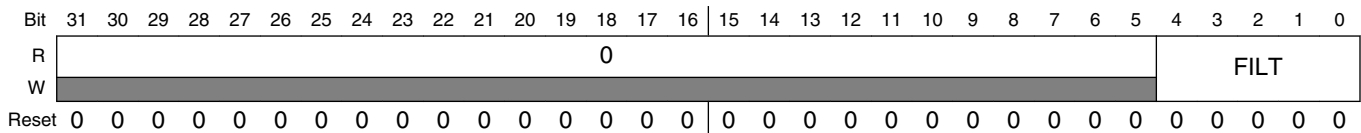
**11.5.7 Digital Filter Width Register (PORTx\_DFWR)**

This register is read only for ports that do not support a digital filter.

The digital filter configuration is valid in all digital pin muxing modes.

## Functional description

Address: Base address + C8h offset



### PORTx\_DFWR field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–0 FILT	Filter Length  The digital filter configuration is valid in all digital pin muxing modes. Configures the maximum size of the glitches, in clock cycles, that the digital filter absorbs for the enabled digital filters. Glitches that are longer than this register setting will pass through the digital filter, and glitches that are equal to or less than this register setting are filtered. Changing the filter length must be done only after all filters are disabled.

## 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
- Drive strength and slew rate configuration
- Open drain enable
- Passive input filter enable
- 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<sup>2</sup>C function is enabled on a pin, that does not override the pullup or open drain 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.

A lock field also exists that allows the configuration for each pin to be locked until the next system reset. When locked, writes to the lower half of that pin control register are ignored, although a bus error is not generated on an attempted write to a locked register.

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 tristated (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. Registers that are locked cannot be written using the global pin control registers.

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:

## Functional description

- 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 or at the output of the digital input filter, if the digital input digital filter is enabled. 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.

### 11.6.4 Digital filter

The digital filter capabilities of the PORT module are available in all digital Pin Muxing modes if the PORT module is enabled.

The clock used for all digital filters within one port can be configured between the bus clock or the 1-kHz LPO clock. This selection must be changed only when all digital filters for that port are disabled. If the digital filters for a port are configured to use the bus clock, then the digital filters are bypassed for the duration of Stop mode. While the digital filters are bypassed, the output of each digital filter always equals the input pin, but the internal state of the digital filters remains static and does not update due to any change on the input pin.

The filter width in clock size is the same for all enabled digital filters within one port and must be changed only when all digital filters for that port are disabled.

The output of each digital filter is logic zero after system reset and whenever a digital filter is disabled. After a digital filter is enabled, the input is synchronized to the filter clock, either the bus clock or the 1-kHz LPO clock. If the synchronized input and the output of the digital filter remain different for a number of filter clock cycles equal to the filter width register configuration, then the output of the digital filter updates to equal the synchronized filter input.

The minimum latency through a digital filter equals two or three filter clock cycles plus the filter width configuration register.





# Chapter 12

## System Integration Module (SIM)

### 12.1 Introduction

#### NOTE

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

The System Integration Module (SIM) provides system control and chip configuration registers.

#### 12.1.1 Features

Features of the SIM include:

- System clocking configuration
  - System clock divide values
  - Architectural clock gating control
- Flash and system RAM size configuration
- FlexTimer external clock, hardware trigger, and fault source selection
- UART0 and UART1 receive/transmit source selection/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 and SIM\_SOPT1CFG registers are located at a different base address than the other SIM registers.

### 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	<a href="#">See section</a>	<a href="#">12.2.1/211</a>
4004_7004	SOPT1 Configuration Register (SIM_SOPT1CFG)	32	R/W	0000_0000h	<a href="#">12.2.2/212</a>
4004_8004	System Options Register 2 (SIM_SOPT2)	32	R/W	0000_1000h	<a href="#">12.2.3/213</a>
4004_800C	System Options Register 4 (SIM_SOPT4)	32	R/W	0000_0000h	<a href="#">12.2.4/215</a>
4004_8010	System Options Register 5 (SIM_SOPT5)	32	R/W	0000_0000h	<a href="#">12.2.5/217</a>
4004_8018	System Options Register 7 (SIM_SOPT7)	32	R/W	0000_0000h	<a href="#">12.2.6/219</a>
4004_801C	System Options Register 8 (SIM_SOPT8)	32	R/W	0000_0000h	<a href="#">12.2.7/220</a>
4004_8024	System Device Identification Register (SIM_SDID)	32	R	<a href="#">See section</a>	<a href="#">12.2.8/222</a>
4004_8034	System Clock Gating Control Register 4 (SIM_SCGC4)	32	R/W	F010_0030h	<a href="#">12.2.9/224</a>
4004_8038	System Clock Gating Control Register 5 (SIM_SCGC5)	32	R/W	0004_0180h	<a href="#">12.2.10/226</a>
4004_803C	System Clock Gating Control Register 6 (SIM_SCGC6)	32	R/W	0000_0001h	<a href="#">12.2.11/227</a>
4004_8040	System Clock Gating Control Register 7 (SIM_SCGC7)	32	R/W	0000_0002h	<a href="#">12.2.12/230</a>
4004_8044	System Clock Divider Register 1 (SIM_CLKDIV1)	32	R/W	<a href="#">See section</a>	<a href="#">12.2.13/230</a>
4004_804C	Flash Configuration Register 1 (SIM_FCFG1)	32	R	<a href="#">See section</a>	<a href="#">12.2.14/233</a>
4004_8050	Flash Configuration Register 2 (SIM_FCFG2)	32	R	<a href="#">See section</a>	<a href="#">12.2.15/235</a>
4004_8054	Unique Identification Register High (SIM_UIDH)	32	R	<a href="#">See section</a>	<a href="#">12.2.16/235</a>
4004_8058	Unique Identification Register Mid-High (SIM_UIDMH)	32	R	<a href="#">See section</a>	<a href="#">12.2.17/236</a>
4004_805C	Unique Identification Register Mid Low (SIM_UIDML)	32	R	<a href="#">See section</a>	<a href="#">12.2.18/236</a>
4004_8060	Unique Identification Register Low (SIM_UIDL)	32	R	<a href="#">See section</a>	<a href="#">12.2.19/237</a>

## 12.2.1 System Options Register 1 (SIM\_SOPT1)

### NOTE

The SOPT1 register is only reset on POR or LVD.

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	Reserved			0								OSC32KSEL		OSC32KOUT		
W	Reserved			Reserved								Reserved		Reserved		
Reset	x*	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	RAMSIZE				0				Reserved							
W	Reserved				Reserved				Reserved							
Reset	x*	x*	x*	x*	0*	0*	0*	0*	0*	0*	x*	x*	x*	x*	x*	x*

\* Notes:

- Reset value loaded during System Reset from Flash IFR.
- x = Undefined at reset.

### SIM\_SOPT1 field descriptions

Field	Description
31–29 Reserved	This field is reserved.
28–20 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 on POR/LVD.  00 System oscillator (OSC32KCLK) 01 Reserved 10 Reserved 11 LPO 1 kHz
17–16 OSC32KOUT	32K Oscillator Clock Output  Outputs the ERCLK32K on the selected pin in all modes of operation (including LLS/VLLS and System Reset), overriding the existing pin mux configuration for that pin. This field is reset only on POR/LVD.  00 ERCLK32K is not output. 01 ERCLK32K is output on PTE0. 10 ERCLK32K is output on PTE26. 11 Reserved.
15–12 RAMSIZE	RAM size  This field specifies the amount of system RAM available on the device.

Table continues on the next page...

**SIM\_SOPT1 field descriptions (continued)**

Field	Description
	0001 8 KB 0011 16 KB 0100 24 KB 0101 32 KB 0110 48 KB 0111 64 KB 1000 96 KB 1001 128 KB 1011 256 KB
11–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–0 Reserved	This field is reserved.

**12.2.2 SOPT1 Configuration Register (SIM\_SOPT1CFG)**

**NOTE**

The SOPT1CFG register is reset on System Reset not VLLS.

Address: 4004\_7000h base + 4h offset = 4004\_7004h

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		0												0		0													
W	0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**SIM\_SOPT1CFG field descriptions**

Field	Description
31–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

### 12.2.3 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PLLFLSEL	
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		TRACECLKSE L	0	0	0	CLKOUTSEL					0	0			
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

#### SIM\_SOPT2 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–16 PLLFLSEL	PLL/FLL clock select Selects the high frequency clock for various peripheral clocking options. <b>NOTE:</b> Not all chips have the option to select a PLL clock.  00 MCGFLLCLK clock

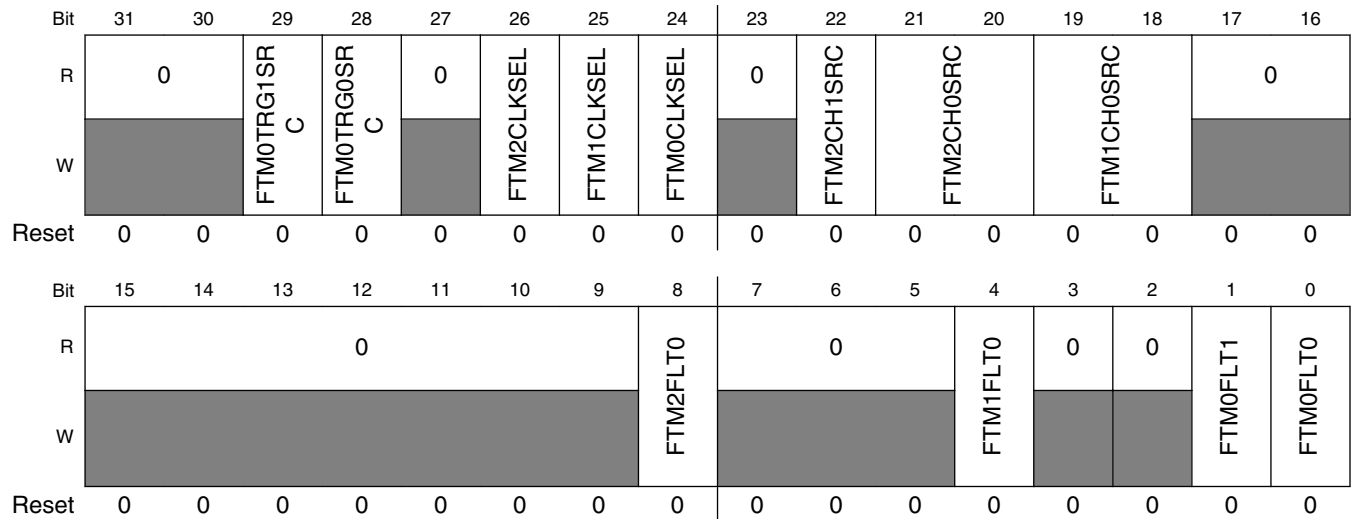
Table continues on the next page...

## SIM\_SOPT2 field descriptions (continued)

Field	Description
	01 Reserved 10 Reserved 11 IRC48 MHz clock
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 TRACECLKSEL	Debug trace clock select Selects the core/system clock or MCG output clock (MCGOUTCLK) as the trace clock source. 0 MCGOUTCLK 1 Core/system clock
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 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 Flash clock 011 LPO clock (1 kHz) 100 MCGIRCLK 101 Reserved 110 OSCERCLK0 111 IRC 48 MHz clock
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

### 12.2.4 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 FTM0TRG1SRC	FlexTimer 0 Hardware Trigger 1 Source Select Selects the source of FTM0 hardware trigger 1. 0 PDB output trigger 1 drives FTM0 hardware trigger 1 1 FTM2 channel match drives FTM0 hardware trigger 1
28 FTM0TRG0SRC	FlexTimer 0 Hardware Trigger 0 Source Select Selects the source of FTM0 hardware trigger 0. 0 HSCMP0 output drives FTM0 hardware trigger 0 1 FTM1 channel match drives FTM0 hardware trigger 0
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FTM2CLKSEL	FlexTimer 2 External Clock Pin Select Selects the external pin used to drive the clock to the FTM2 module. <b>NOTE:</b> The selected pin must also be configured for the FTM2 module external clock function through the appropriate pin control register in the port control module. 0 FTM2 external clock driven by FTM_CLK0 pin. 1 FTM2 external clock driven by FTM_CLK1 pin.
25 FTM1CLKSEL	FTM1 External Clock Pin Select Selects the external pin used to drive the clock to the FTM1 module.

Table continues on the next page...

## SIM\_SOPT4 field descriptions (continued)

Field	Description
	<p><b>NOTE:</b> The selected pin must also be configured for the FTM external clock function through the appropriate pin control register in the port control module.</p> <p>0 FTM_CLK0 pin 1 FTM_CLK1 pin</p>
24 FTMOCLKSEL	<p>FlexTimer 0 External Clock Pin Select</p> <p>Selects the external pin used to drive the clock to the FTM0 module.</p> <p><b>NOTE:</b> The selected pin must also be configured for the FTM external clock function through the appropriate pin control register in the port control module.</p> <p>0 FTM_CLK0 pin 1 FTM_CLK1 pin</p>
23 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
22 FTM2CH1SRC	<p>FTM2 channel 1 input capture source select</p> <p>0 FTM2_CH1 signal 1 Exclusive OR of FTM2_CH1, FTM2_CH0 and FTM1_CH1.</p>
21–20 FTM2CH0SRC	<p>FTM2 channel 0 input capture source select</p> <p>Selects the source for FTM2 channel 0 input capture.</p> <p><b>NOTE:</b> When the FTM is not in input capture mode, clear this field.</p> <p>00 FTM2_CH0 signal 01 CMP0 output 10 CMP1 output 11 Reserved</p>
19–18 FTM1CH0SRC	<p>FTM1 channel 0 input capture source select</p> <p>Selects the source for FTM1 channel 0 input capture.</p> <p><b>NOTE:</b> When the FTM is not in input capture mode, clear this field.</p> <p>00 FTM1_CH0 signal 01 CMP0 output 10 CMP1 output 11 Reserved</p>
17–9 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
8 FTM2FLT0	<p>FTM2 Fault 0 Select</p> <p>Selects the source of FTM2 fault 0.</p> <p><b>NOTE:</b> The pin source for fault 0 must be configured for the FTM module fault function through the appropriate PORTx pin control register.</p> <p>0 FTM2_FLT0 pin 1 CMP0 out</p>

Table continues on the next page...



## SIM\_SOPT4 field descriptions (continued)

Field	Description
7-5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 FTM1FLT0	FTM1 Fault 0 Select Selects the source of FTM1 fault 0.  <b>NOTE:</b> 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
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 FTM0FLT1	FTM0 Fault 1 Select Selects the source of FTM0 fault 1.  <b>NOTE:</b> 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.  <b>NOTE:</b> 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

## 12.2.5 System Options Register 5 (SIM\_SOPT5)

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

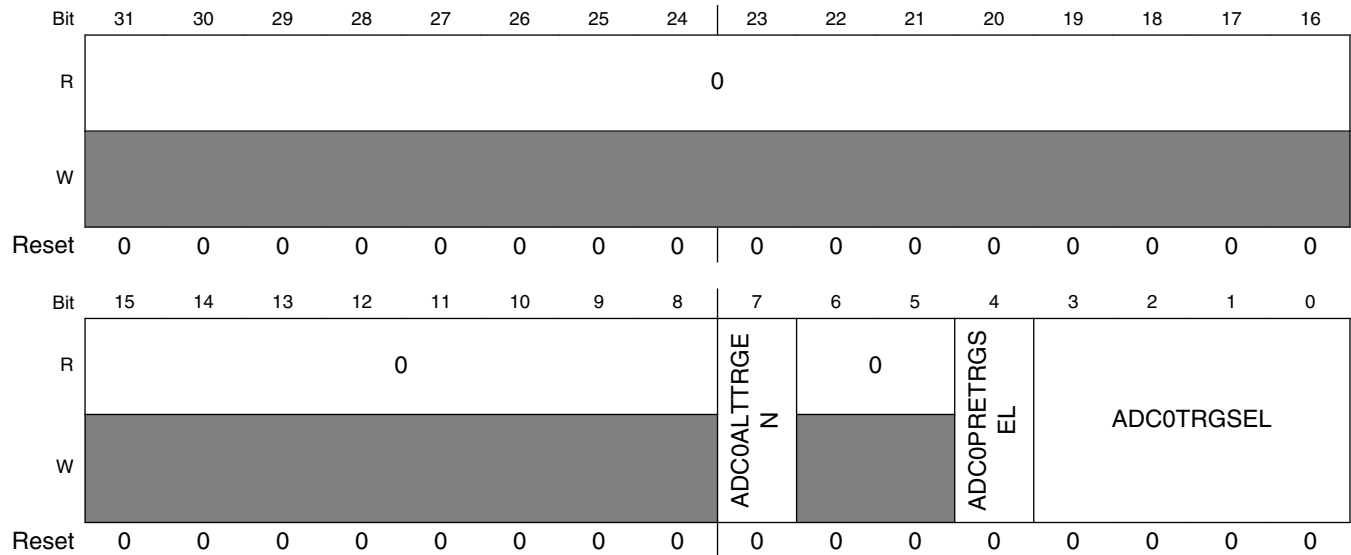
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															0
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								UART1RXSR	UART1TXSR	UART0RXSR	UART0TXSR				
W	0								C	C	C	C				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## SIM\_SOPT5 field descriptions

Field	Description
31–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 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
1–0 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

## 12.2.6 System Options Register 7 (SIM\_SOPT7)

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



### SIM\_SOPT7 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADC0ALTTTRGEN	ADC0 alternate trigger enable Enable alternative conversion triggers for ADC0.  0 PDB trigger selected for ADC0. 1 Alternate trigger selected for ADC0.
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ADC0PRETRGSEL	ADC0 pretrigger select Selects the ADC0 pre-trigger source when alternative triggers are enabled through ADC0ALTTTRGEN.  0 Pre-trigger A 1 Pre-trigger B
3–0 ADC0TRGSEL	ADC0 trigger select Selects the ADC0 trigger source when alternative triggers are functional in stop and VLPS modes. .  0000 PDB external trigger pin input (PDB0_EXTRG) 0001 High speed comparator 0 output 0010 High speed comparator 1 output 0011 Reserved

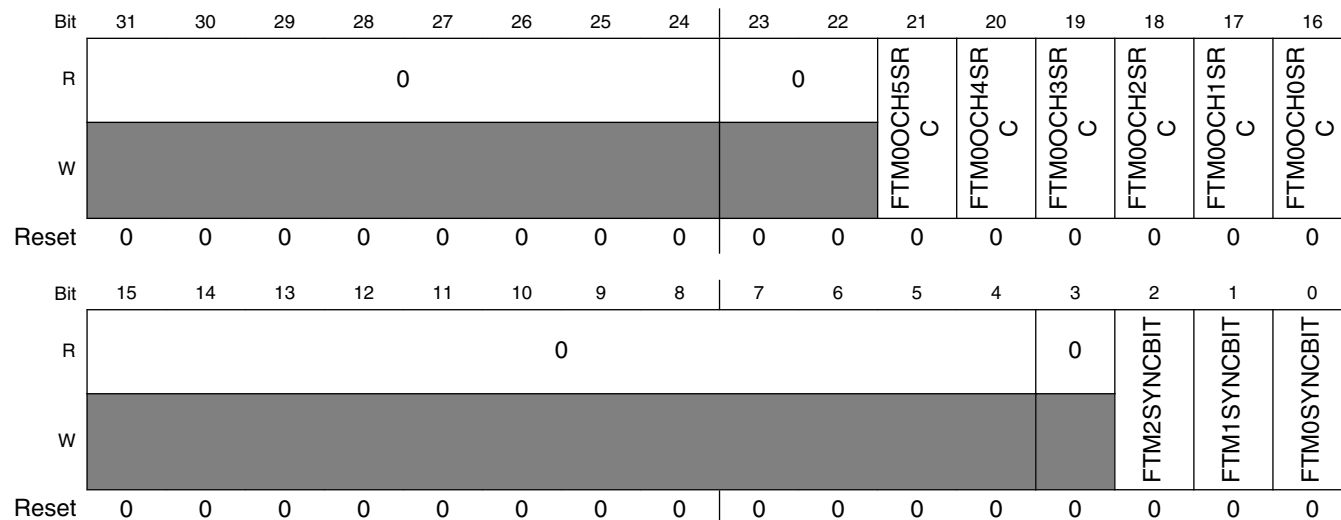
Table continues on the next page...

### SIM\_SOPT7 field descriptions (continued)

Field	Description
0100	PIT trigger 0
0101	PIT trigger 1
0110	PIT trigger 2
0111	PIT trigger 3
1000	FTM0 trigger
1001	FTM1 trigger
1010	FTM2 trigger
1011	Reserved
1100	Reserved
1101	Reserved
1110	Low-power timer (LPTMR) trigger
1111	Reserved

## 12.2.7 System Options Register 8 (SIM\_SOPT8)

Address: 4004\_7000h base + 101Ch offset = 4004\_801Ch



### 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.
23–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 FTM0CH5SRC	FTM0 channel 5 output source 0 FTM0_CH5 pin is output of FTM0 channel 5 output 1 FTM0_CH5 pin is output of FTM0 channel 5 output, modulated by FTM1 channel 1 output

Table continues on the next page...

**SIM\_SOPT8 field descriptions (continued)**

<b>Field</b>	<b>Description</b>
20 FTM0OCH4SRC	FTM0 channel 4 output source 0 FTM0_CH4 pin is output of FTM0 channel 4 output 1 FTM0_CH4 pin is output of FTM0 channel 4 output, modulated by FTM1 channel 1 output
19 FTM0OCH3SRC	FTM0 channel 3 output source 0 FTM0_CH3 pin is output of FTM0 channel 3 output 1 FTM0_CH3 pin is output of FTM0 channel 3 output, modulated by FTM1 channel 1 output
18 FTM0OCH2SRC	FTM0 channel 2 output source 0 FTM0_CH2 pin is output of FTM0 channel 2 output 1 FTM0_CH2 pin is output of FTM0 channel 2 output, modulated by FTM1 channel 1 output
17 FTM0OCH1SRC	FTM0 channel 1 output source 0 FTM0_CH1 pin is output of FTM0 channel 1 output 1 FTM0_CH1 pin is output of FTM0 channel 1 output, modulated by FTM1 channel 1 output
16 FTM0OCH0SRC	FTM0 channel 0 output source 0 FTM0_CH0 pin is output of FTM0 channel 0 output 1 FTM0_CH0 pin is output of FTM0 channel 0 output, modulated by FTM1 channel 1 output
15–4 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 FTM2SYNCSBIT	FTM2 Hardware Trigger 0 Software Synchronization 0 No effect. 1 Write 1 to assert the TRIG0 input to FTM2, software must clear this bit to allow other trigger sources to assert.
1 FTM1SYNCSBIT	FTM1 Hardware Trigger 0 Software Synchronization 0 No effect. 1 Write 1 to assert the TRIG0 input to FTM1, software must clear this bit to allow other trigger sources to assert.
0 FTM0SYNCSBIT	FTM0 Hardware Trigger 0 Software Synchronization 0 No effect 1 Write 1 to assert the TRIG0 input to FTM0, software must clear this bit to allow other trigger sources to assert.

## 12.2.8 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	FAMILYID				SUBFAMID				SERIESID				0				REVID				DIEID				FAMID				PINID			
W	[Shaded]																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	0	0	0	0	x*	x*	x*	x*	0	0	1	0	1	x*	x*	x*	x*	x*	x*	x*

\* Notes:

- x = Undefined at reset.

### SIM\_SDID field descriptions

Field	Description
31–28 FAMILYID	<p>Kinetis Family ID</p> <p>Specifies the Kinetis family of the device.</p> <p>0000 K0x Family 0001 K1x Family 0010 K2x Family 0011 K3x Family 0100 K4x Family 0110 K6x Family 0111 K7x Family</p>
27–24 SUBFAMID	<p>Kinetis Sub-Family ID</p> <p>Specifies the Kinetis sub-family of the device.</p> <p>0000 Kx0 Subfamily 0001 Kx1 Subfamily (tamper detect) 0010 Kx2 Subfamily 0011 Kx3 Subfamily (tamper detect) 0100 Kx4 Subfamily 0101 Kx5 Subfamily (tamper detect) 0110 Kx6 Subfamily</p>
23–20 SERIESID	<p>Kinetis Series ID</p> <p>Specifies the Kinetis series of the device.</p> <p>0000 Kinetis K series 0001 Kinetis L series 0101 Kinetis W series 0110 Kinetis V series</p>
19–16 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
15–12 REVID	<p>Device revision number</p> <p>Specifies the silicon implementation number for the device.</p>

Table continues on the next page...

**SIM\_SDID field descriptions (continued)**

<b>Field</b>	<b>Description</b>
11–7 DIEID	Device Die ID Specifies the silicon feature set identification number for the device.
6–4 FAMID	Kinetis family identification This field is maintained for compatibility only, but has been superceded by the SERIESID, FAMILYID and SUBFAMID fields in this register.  000 K1x Family (without tamper) 001 K2x Family (without tamper) 010 K3x Family or K1x/K6x Family (with tamper) 011 K4x Family or K2x Family (with tamper) 100 K6x Family (without tamper) 101 K7x Family 110 Reserved 111 Reserved
3–0 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 80-pin 0111 81-pin or 121-pin 1000 100-pin 1001 121-pin 1010 144-pin 1011 Custom pinout (WLCSP) 1100 169-pin 1101 Reserved 1110 256-pin 1111 Reserved

## 12.2.9 System Clock Gating Control Register 4 (SIM\_SCGC4)

Address: 4004\_7000h base + 1034h offset = 4004\_8034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	1				0								VREF	CMP	0	0
W	[Reserved]															
Reset	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	UART1	UART0	0	0	I2C0	1	0	0	EWM	0			
W	[Reserved]															
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

### 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.
27–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 VREF	VREF Clock Gate Control  This bit controls the clock gate to the VREF module.  0 Clock disabled 1 Clock enabled
19 CMP	Comparator Clock Gate Control  This bit controls the clock gate to the comparator module.  0 Clock disabled 1 Clock enabled
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–14 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 UART1	UART1 Clock Gate Control  This bit controls the clock gate to the UART1 module.

Table continues on the next page...

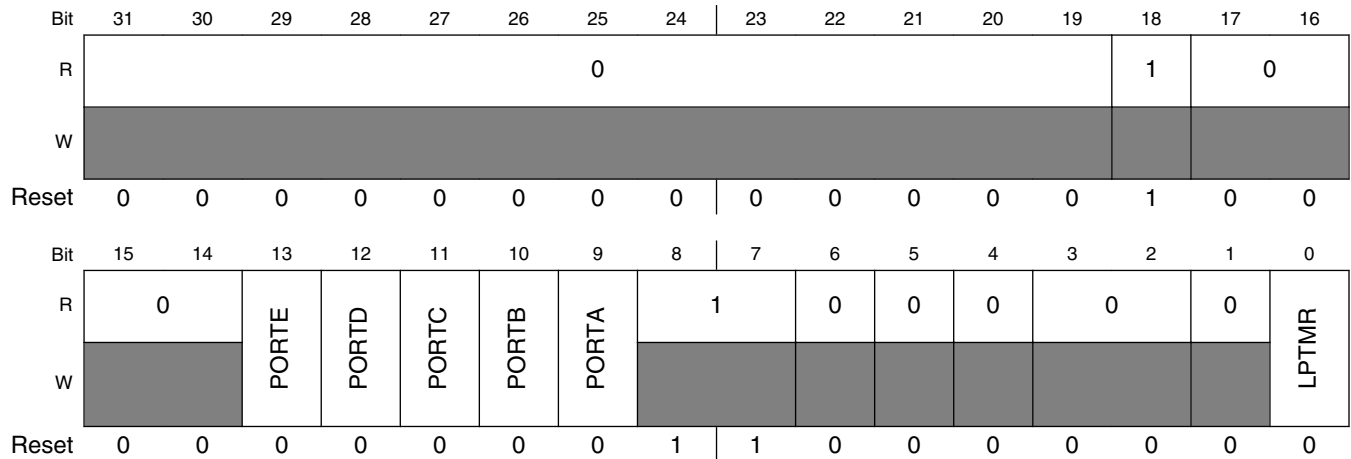


**SIM\_SCGC4 field descriptions (continued)**

<b>Field</b>	<b>Description</b>
	0 Clock disabled 1 Clock enabled
10 UART0	UART0 Clock Gate Control  This bit controls the clock gate to the UART0 module.  0 Clock disabled 1 Clock enabled
9–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 I2C0	I2C0 Clock Gate Control  This bit controls the clock gate to the I <sup>2</sup> C0 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 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 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.

### 12.2.10 System Clock Gating Control Register 5 (SIM\_SCGC5)

Address: 4004\_7000h base + 1038h offset = 4004\_8038h



**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.
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  This bit controls the clock gate to the Port E module.  0 Clock disabled 1 Clock enabled
12 PORTD	Port D Clock Gate Control  This bit controls the clock gate to the Port D module.  0 Clock disabled 1 Clock enabled
11 PORTC	Port C Clock Gate Control  This bit controls the clock gate to the Port C module.  0 Clock disabled 1 Clock enabled
10 PORTB	Port B Clock Gate Control  This bit controls the clock gate to the Port B module.

*Table continues on the next page...*

## SIM\_SCGC5 field descriptions (continued)

Field	Description
	0 Clock disabled 1 Clock enabled
9 PORTA	Port A Clock Gate Control  This bit 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 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 LPTMR	Low Power Timer Access Control  This bit controls software access to the Low Power Timer module.  0 Access disabled 1 Access enabled

## 12.2.11 System Clock Gating Control Register 6 (SIM\_SCGC6)

Address: 4004\_7000h base + 103Ch offset = 4004\_803Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		0	0	0							0	0				0
W	DAC0				ADC0	FTM2	FTM1	FTM0	PIT	PDB				CRC		
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	0	0		0	0	0	0	0	0	0	0	0			
W				SPI0											DMAMUX	FTF
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

## SIM\_SCGC6 field descriptions

Field	Description
31 DAC0	DAC0 Clock Gate Control  This bit controls the clock gate to the DAC0 module.  0 Clock disabled 1 Clock enabled
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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 ADC0	ADC0 Clock Gate Control  This bit controls the clock gate to the ADC0 module.  0 Clock disabled 1 Clock enabled
26 FTM2	FTM2 Clock Gate Control  This bit controls the clock gate to the FTM2 module.  0 Clock disabled 1 Clock enabled
25 FTM1	FTM1 Clock Gate Control  This bit controls the clock gate to the FTM1 module.  0 Clock disabled 1 Clock enabled
24 FTM0	FTM0 Clock Gate Control  This bit controls the clock gate to the FTM0 module.  0 Clock disabled 1 Clock enabled
23 PIT	PIT Clock Gate Control  This bit controls the clock gate to the PIT module.  0 Clock disabled 1 Clock enabled
22 PDB	PDB Clock Gate Control  This bit controls the clock gate to the PDB module.  0 Clock disabled 1 Clock enabled
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

*Table continues on the next page...*

**SIM\_SCGC6 field descriptions (continued)**

<b>Field</b>	<b>Description</b>
20–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 CRC	CRC Clock Gate Control  This bit controls the clock gate to the CRC module.  0 Clock disabled 1 Clock enabled
17–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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 SPI0	SPI0 Clock Gate Control  This bit controls the clock gate to the SPI0 module.  0 Clock disabled 1 Clock 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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DMAMUX	DMA Mux Clock Gate Control  This bit controls the clock gate to the DMA Mux module.  0 Clock disabled 1 Clock enabled
0 FTF	Flash Memory Clock Gate Control

*Table continues on the next page...*

**SIM\_SCGC6 field descriptions (continued)**

Field	Description
	This bit 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 and HSRUN mode is blocked.
0	Clock disabled
1	Clock enabled

**12.2.12 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											0	0	DMA	0		
W																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	1	0

**SIM\_SCGC7 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	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 DMA	DMA Clock Gate Control  This bit controls the clock gate to the DMA 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.

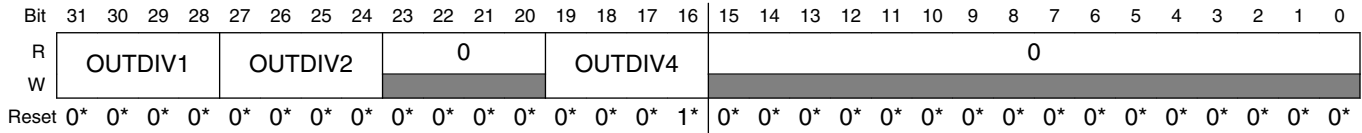
**12.2.13 System Clock Divider Register 1 (SIM\_CLKDIV1)**

When updating CLKDIV1, update all fields using the one write command. Attempting to write an invalid clock ratio to the CLKDIV1 register will cause the write to be ignored. The maximum divide ratio that can be programmed between core/system clock and the other divided clocks is divide by 8. When OUTDIV1 equals 0000 (divide by 1), the other dividers cannot be set higher than 0111 (divide by 8).

**NOTE**

The CLKDIV1 register cannot be written to when the device is in VLPR mode.

Address: 4004\_7000h base + 1044h offset = 4004\_8044h



\* Notes:

- Reset value loaded during System Reset from FTF\_FOFT[LPBOOT].

**SIM\_CLKDIV1 field descriptions**

Field	Description
31–28 OUTDIV1	<p>Clock 1 output divider value</p> <p>This field sets the divide value for the core/system clock from MCGOUTCLK. At the end of reset, it is loaded with either 0000 or 0111 depending on FTF_FOFT[LPBOOT].</p> <p>0000 Divide-by-1.                      0001 Divide-by-2.                      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.</p>
27–24 OUTDIV2	<p>Clock 2 output divider value</p> <p>This field sets the divide value for the bus clock from MCGOUTCLK. At the end of reset, it is loaded with either 0000 or 0111 depending on FTF_FOFT[LPBOOT]. The bus clock frequency must be an integer divide of the core/system clock frequency.</p> <p>0000 Divide-by-1.                      0001 Divide-by-2.                      0010 Divide-by-3.                      0011 Divide-by-4.                      0100 Divide-by-5.                      0101 Divide-by-6.                      0110 Divide-by-7.                      0111 Divide-by-8.</p>

Table continues on the next page...

## SIM\_CLKDIV1 field descriptions (continued)

Field	Description
	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.
23–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–16 OUTDIV4	Clock 4 output divider value  This field sets the divide value for the flash clock from MCGOUTCLK. At the end of reset, it is loaded with either 0001 or 1111 depending on FTF_FOPT[LPBOOT]. The flash clock frequency must be an integer divide of the system clock frequency.  0000 Divide-by-1. 0001 Divide-by-2. 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.
15–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.



### 12.2.14 Flash Configuration Register 1 (SIM\_FCFG1)

Address: 4004\_7000h base + 104Ch offset = 4004\_804Ch

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

\* Notes:

- Reset value loaded during System Reset from Flash IFR.

#### 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 . Undefined values are reserved.  0011 32 KB of program flash memory 0101 64 KB of program flash memory 0111 128 KB of program flash memory 1001 256 KB of program flash memory

Table continues on the next page...

**SIM\_FCFG1 field descriptions (continued)**

Field	Description
	1011 512 KB of program flash memory 1101 1024 KB of program flash memory 1111 128 KB of program flash memory
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 1.
15–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
7–2 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 Wait 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 bit should be clear during VLP modes. The Flash will be automatically enabled again at the end of Wait mode so interrupt vectors do not need to be relocated out of Flash memory. The wakeup time from Wait mode is extended when this bit is set.  0 Flash remains enabled during Wait mode 1 Flash is disabled for the duration of Wait 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 bit 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

### 12.2.15 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	MAXADDR0							1	Reserved						
W	[Greyed out]															
Reset	0*	1*	1*	1*	1*	1*	1*	1*	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	[Greyed out]															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

\* Notes:

- Reset value partially loaded during System Reset from Flash IFR.

#### 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 MAXADDR0	Max address block 0  This field concatenated with 13 trailing zeros indicates the first invalid address of each program flash block.  For example, if MAXADDR0 = 0x20 the first invalid address of flash block 0 is 0x0004_0000. This would be the MAXADDR0 value for a device with 256 KB program flash in flash block 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
22–16 Reserved	This field is reserved.
15–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

### 12.2.16 Unique Identification Register High (SIM\_UIDH)

Address: 4004\_7000h base + 1054h offset = 4004\_8054h

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	UID																															
W	[Greyed out]																															
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*	

\* Notes:

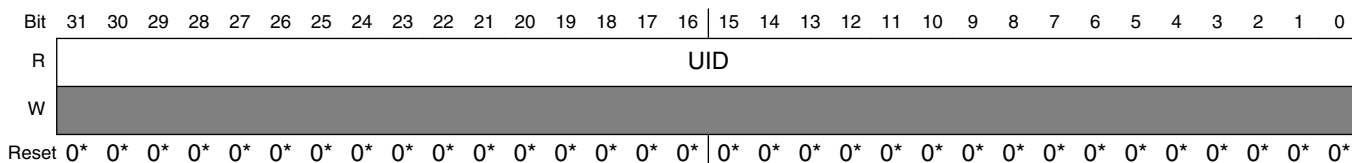
- Reset value loaded during System Reset from Flash IFR.

### SIM\_UIDH field descriptions

Field	Description
31–0 UID	Unique Identification Unique identification for the device.

## 12.2.17 Unique Identification Register Mid-High (SIM\_UIDMH)

Address: 4004\_7000h base + 1058h offset = 4004\_8058h



\* Notes:

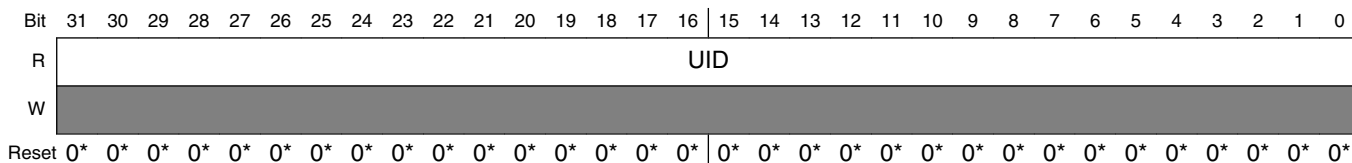
- Reset value loaded during System Reset from Flash IFR.

### SIM\_UIDMH field descriptions

Field	Description
31–0 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:

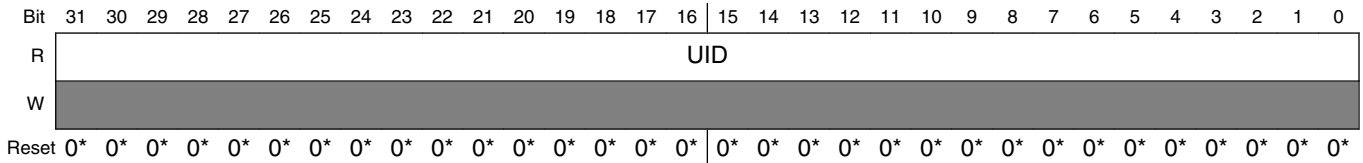
- Reset value loaded during System Reset from Flash IFR.

### SIM\_UIDML field descriptions

Field	Description
31–0 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:

- Reset value loaded during System Reset from Flash IFR.

### SIM\_UIDL field descriptions

Field	Description
31–0 UID	Unique Identification Unique identification for the device.

## 12.3 Functional description

For more information about the functions of SIM, see the [Introduction](#) section.



# Chapter 13

## Kinetis Flashloader

### 13.1 Chip-Specific Information

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

**Table 13-1. Kinetis Flashloader Peripheral Pinmux**

Port	Signal
PTE0	UART1_TX
PTE1	UART1_RX
PTC10	I2C1_SCL
PTC11	I2C1_SDA
PTD4	SPI1_SS_b
PTD6	SPI1_MISO
PTD7	SPI1_MOSI
PTD5	SPI1_SCK

### 13.2 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 the device, the Kinetis Flashloader can interface with UART, 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, I2C, and SPI peripheral interfaces
- Automatic detection of the active peripheral
- UART peripheral implements 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 13-2. 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	
FlashEraseAll	Erase the entire flash array	
FlashEraseRegion	Erase a range of sectors in flash	
FlashProgramOnce	Writes data provided in a command packet to a specified range of bytes in the program once field	
FlashReadOnce	Returns the contents of the program once field by given index and byte count	
FlashReadResource	Returns the contents of the IFR field or Flash firmware ID, by given offset, byte count and option	
WriteMemory	Write data to memory	
ReadMemory	Read data from memory	

*Table continues on the next page...*



**Table 13-2. Commands supported by the Kinetis Flashloader (continued)**

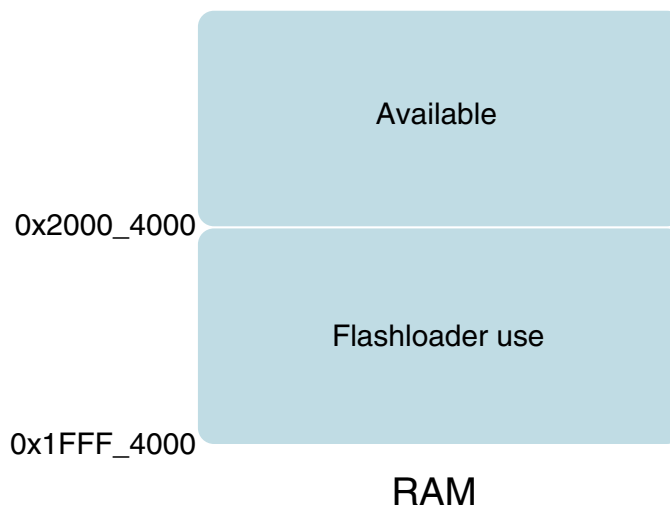
Command	Description	When flash security is enabled, then this command is
FlashSecurityDisable	Attempt to unlock flash security using the backdoor key	Supported
GetProperty	Get the current value of a property	
Reset	Reset the chip	
SetProperty	Attempt to modify a writable property	Supported

## 13.3 Functional Description

The following sub-sections describe the Kinetis Flashloader functionality.

### 13.3.1 Memory Maps

While executing, the Kinetis Flashloader uses RAM memory.



**Figure 13-1. Kinetis Flashloader RAM Memory Map**

### 13.3.2 Kinetis Flashloader

### 13.3.3 Start-up Process

As the Kinetis Flashloader begins executing, flashloader operations begin:

1. The flashloader's .data and .bss sections are 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.

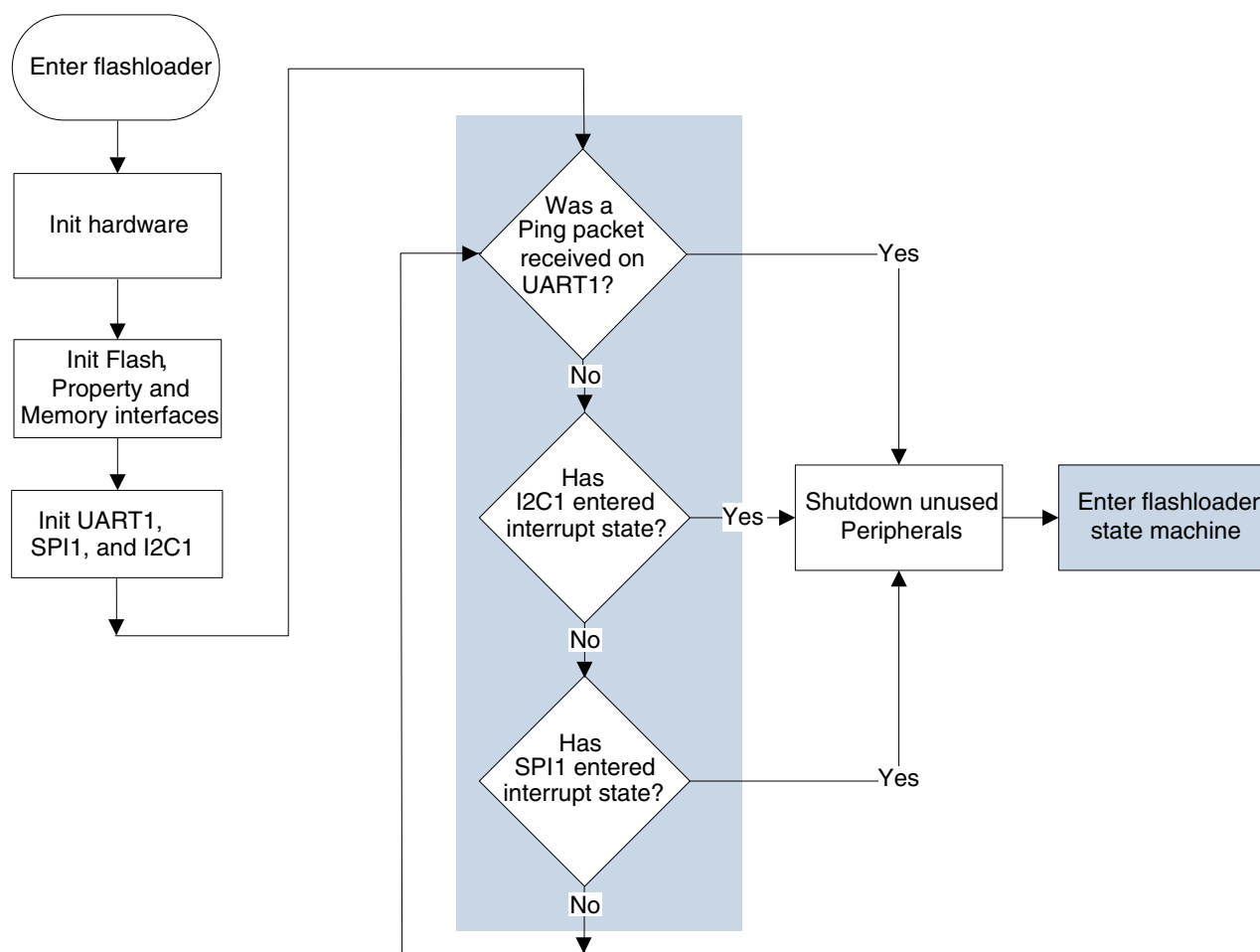


Figure 13-2. Kinetis Flashloader Start-up Flowchart

### 13.3.4 Clock Configuration

The Kinetis Flashloader uses the clock configuration of the chip out of reset. The flashloader does not modify any clocks.

### 13.3.5 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 it (the data phase) is part of the **original command**.
- If the data phase is **outgoing** (from flashloader to host), then it (the data phase) is part of the **response command**.

#### 13.3.5.1 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)

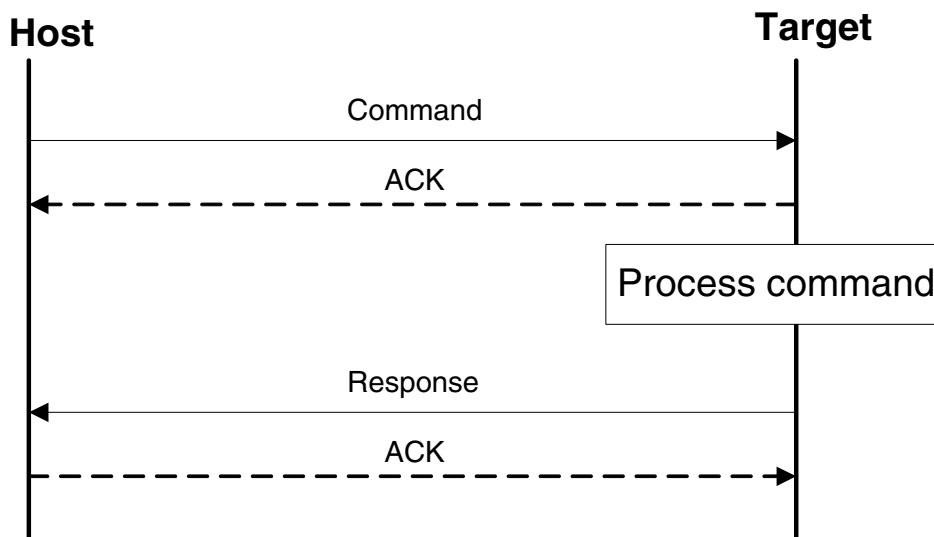


Figure 13-3. Command with No Data Phase

### 13.3.5.2 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)

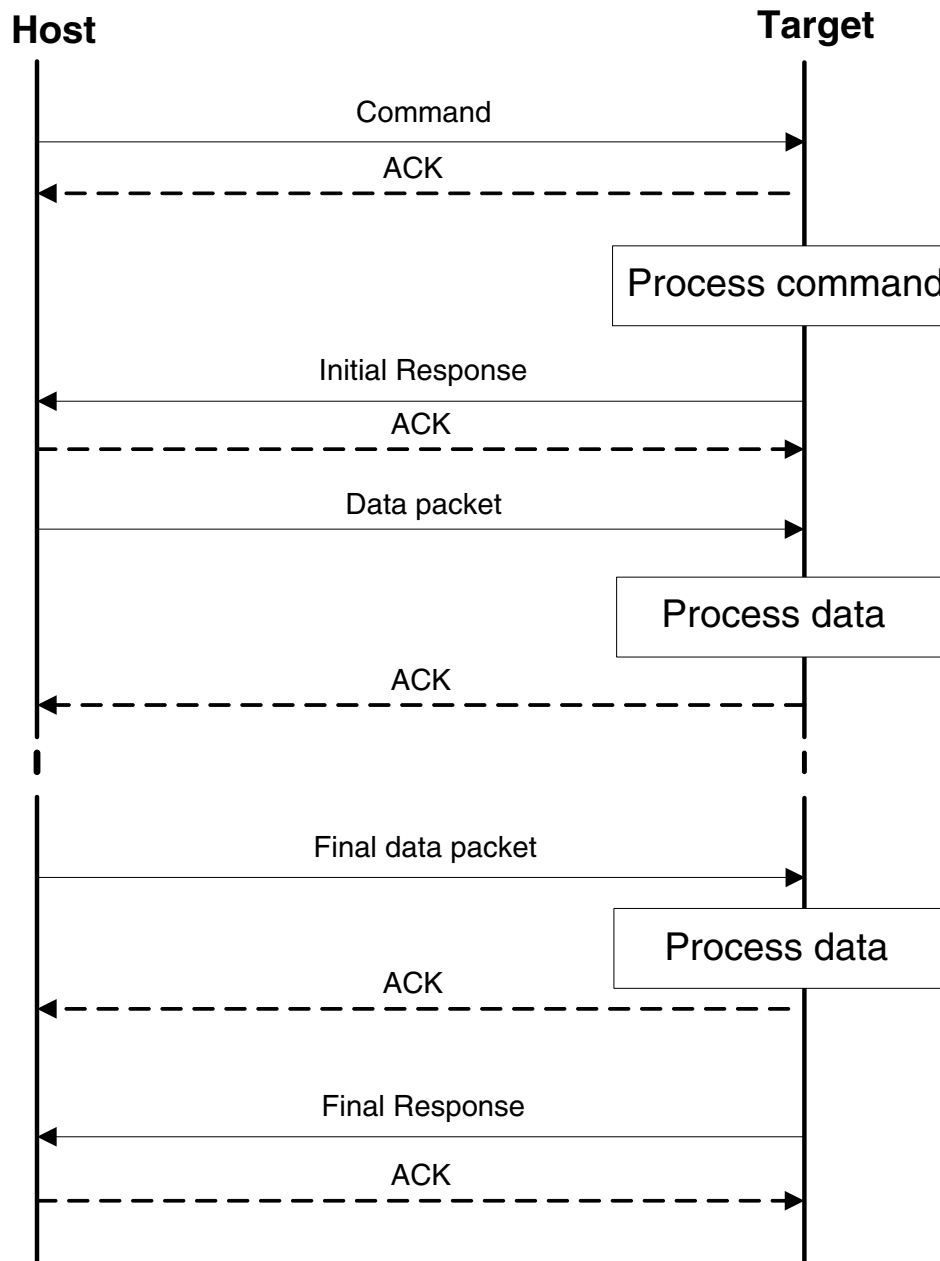


Figure 13-4. Command with incoming data phase

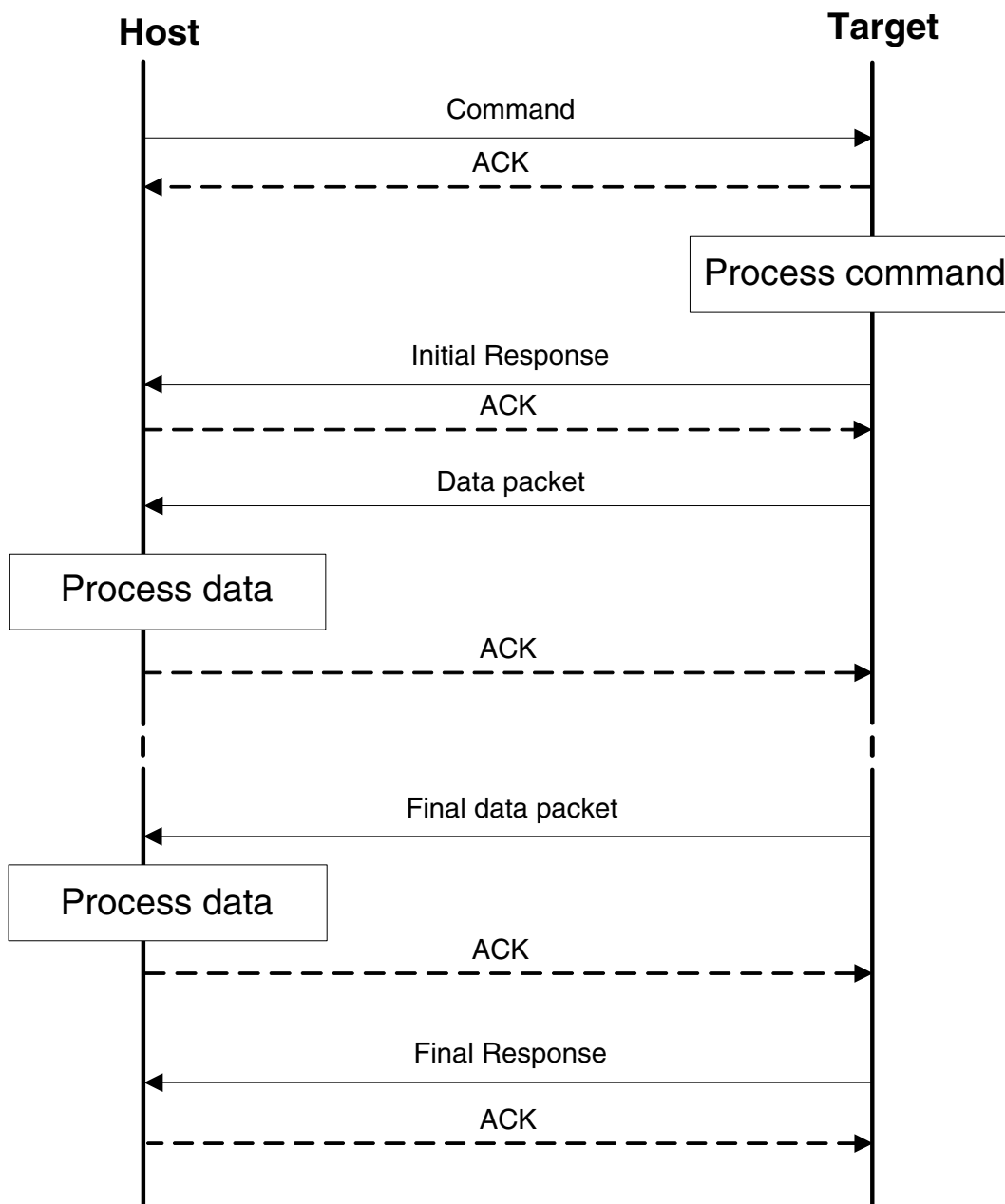
**NOTE**

- 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.

**13.3.5.3 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.

### 13.3.6 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 device:

- Ping packet
- Ping Response packet
- Framing packet
- Command packet
- Data packet
- Response packet

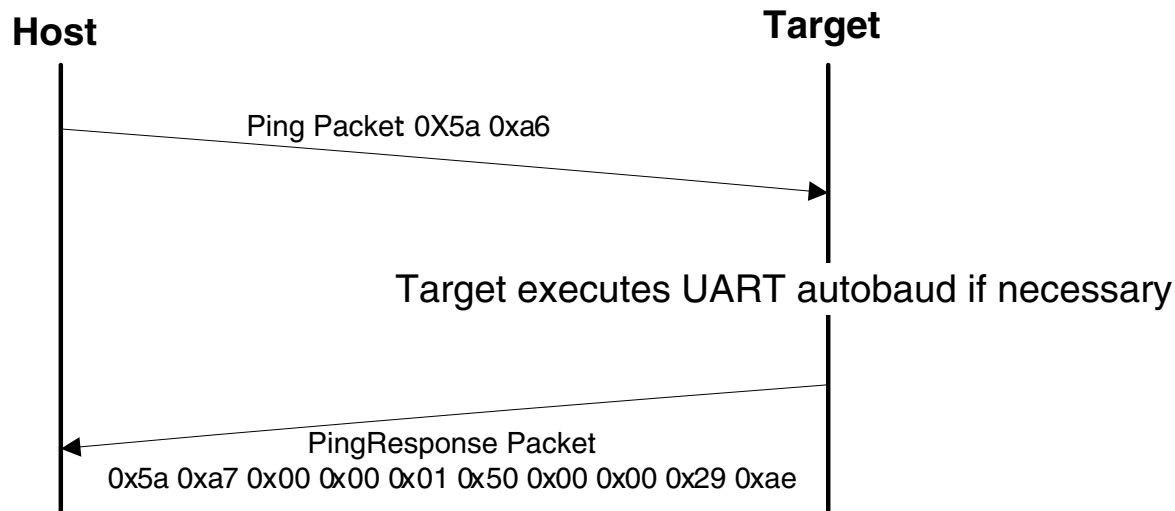
All fields in the packets are in little-endian byte order.

#### 13.3.6.1 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 13-3. Ping Packet Format**

Byte #	Value	Name
0	0x5A	start byte
1	0xA6	ping



**Figure 13-6. Ping Packet Protocol Sequence**

### 13.3.6.2 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 13-4. 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

The Serial Protocol Version number returned is 1.1.0.



### 13.3.6.3 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.

**Table 13-5. 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	

#### CRC16 algorithm:

```
uint16_t crc16_update(const uint8_t * src, uint32_t lengthInBytes)
{
    uint32_t crc = 0;
    uint32_t j;
    for (j=0; j < lengthInBytes; ++j)
    {
        uint32_t i;
        uint32_t byte = src[j];
        crc ^= byte << 8;
        for (i = 0; i < 8; ++i)
        {
            uint32_t temp = crc << 1;
            if (crc & 0x8000)
            {
                temp ^= 0x1021;
            }
            crc = temp;
        }
    }
}
```

## Functional Description

```

}
return crc;
}

```

A special framing packet that contains only a start byte and a packet type is used for synchronization between the host and target.

**Table 13-6. 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 13-7. 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.
0xA7	kFramingPacketType_PingResponse	A response to Ping; contains the framing protocol version number and options.

### 13.3.6.4 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

**Table 13-8. 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 13-9. 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 13-10. Command Tags**

Command Tag	Name	
0x01	FlashEraseAll	The command tag specifies one of the commands supported by the Kinetis Flashloader. The valid command tags for the Kinetis Flashloader are listed here.
0x02	FlashEraseRegion	
0x03	ReadMemory	
0x04	WriteMemory	
0x06	FlashSecurityDisable	
0x07	GetProperty	
0x09	Execute	
0x0B	Reset	
0x0C	SetProperty	
0x0D	Reserved	

**Table 13-11. Response Tags**

Response Tag	Name	
0xA0	GenericResponse	The response tag specifies one of the responses the Kinetis Flashloader (target) returns to the host. The valid response tags are listed here.
0xA7	GetPropertyResponse (used for sending responses to GetProperty command only)	
0xA3	ReadMemoryResponse (used for sending responses to ReadMemory command only)	



**Table 13-13. 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. <a href="#">Table 13-39</a> , 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 13-14. 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.

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 13-15. 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.

### 13.3.7 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 13-2](#), Commands supported.
- For a list of status codes returned by the Kinetis Flashloader, see [Table 13-39](#), Kinetis Flashloader Status Error Codes.

#### 13.3.7.1 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 13-35](#).

The 32-bit property tag is the only parameter required for GetProperty command.

**Table 13-16. Parameters for GetProperty Command**

Byte #	Command
0 - 3	Property tag

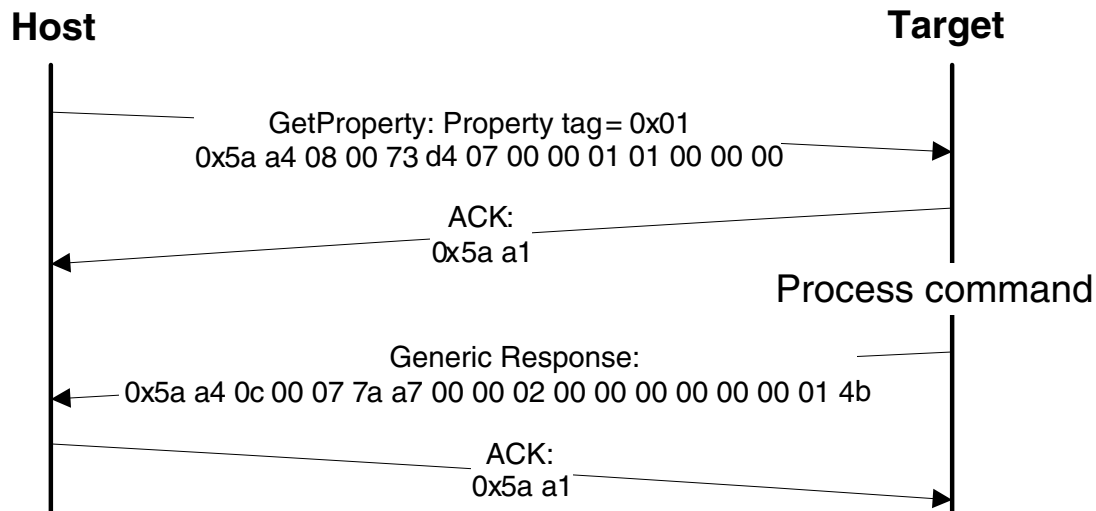


Figure 13-7. Protocol Sequence for GetProperty Command

Table 13-17. GetProperty Command Packet Format (Example)

GetProperty	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x08 0x00
	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 13-18. 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

### 13.3.7.2 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 13-35](#), 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 13-19. Parameters for SetProperty Command**

Byte #	Command
0 - 3	Property tag
4 - 7	Property value



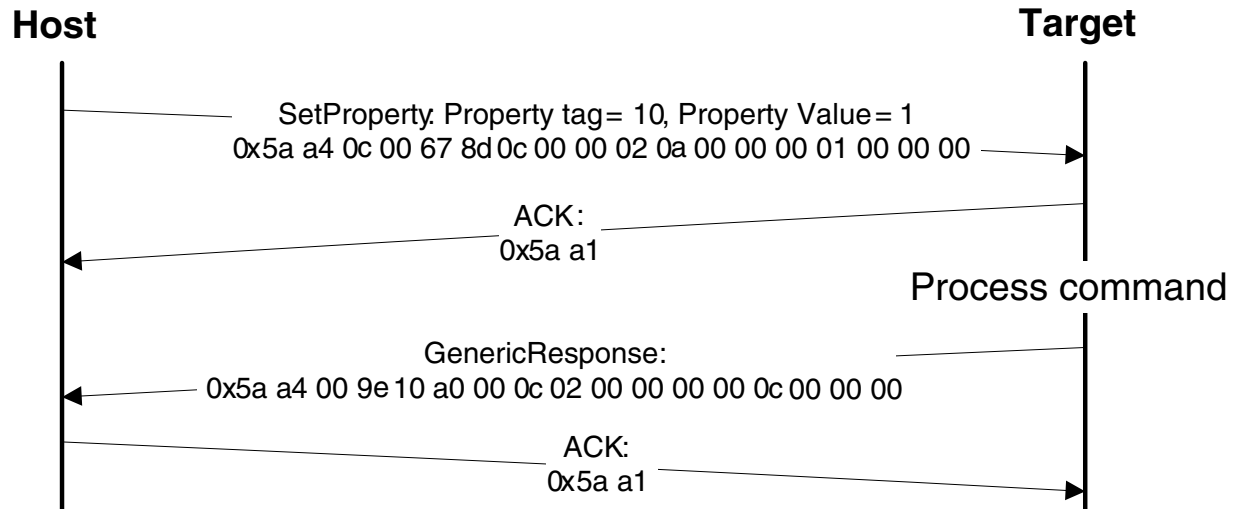


Figure 13-8. Protocol Sequence for SetProperty Command

Table 13-20. 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 13-21. SetProperty Response Status Codes

Status Code
kStatus_Success
kStatus_ReadOnly
kStatus_UnknownProperty
kStatus_InvalidArgument

### 13.3.7.3 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.

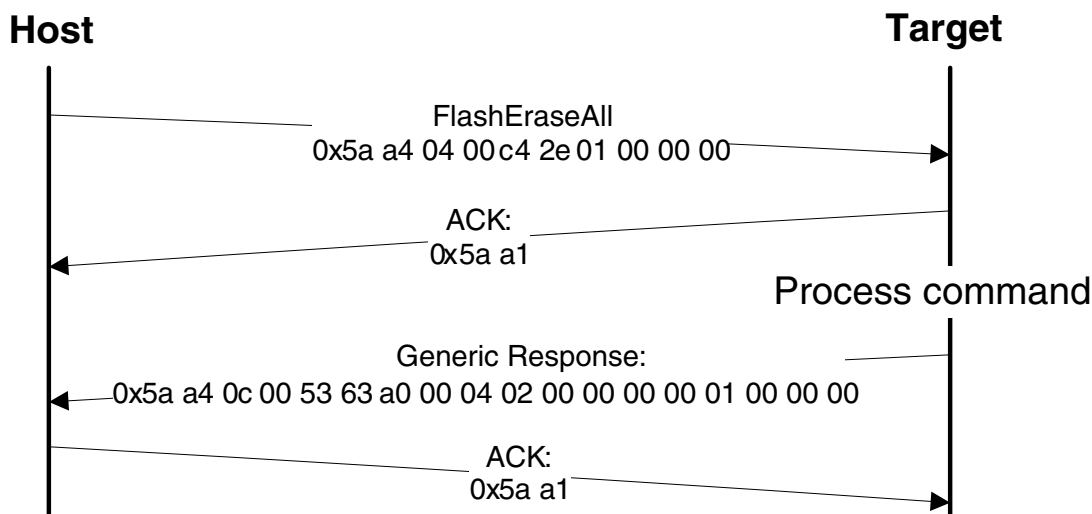


Figure 13-9. Protocol Sequence for FlashEraseAll Command

Table 13-22. 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

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.

### 13.3.7.4 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory.

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(101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command will fail and return kStatus\_FlashAddressError(102). If any part of the region specified is protected, the FlashEraseRegion command will fail and return kStatus\_MemoryRangeInvalid(10200).

**Table 13-23. Parameters for FlashEraseRegion Command**

Byte #	Parameter
0 - 3	Start address
4 - 7	Byte count

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 13-24. FlashEraseRegion Response Status Codes**

Status Code
kStatus_Success (0)
kStatus_MemoryRangeInvalid (10200)
kStatus_FlashAlignmentError (101)
kStatus_FlashAddressError (102)
kStatus_FlashAccessError (103)
kStatus_FlashProtectionViolation (104)
kStatus_FlashCommandFailure (105)

### 13.3.7.5 FillMemory command

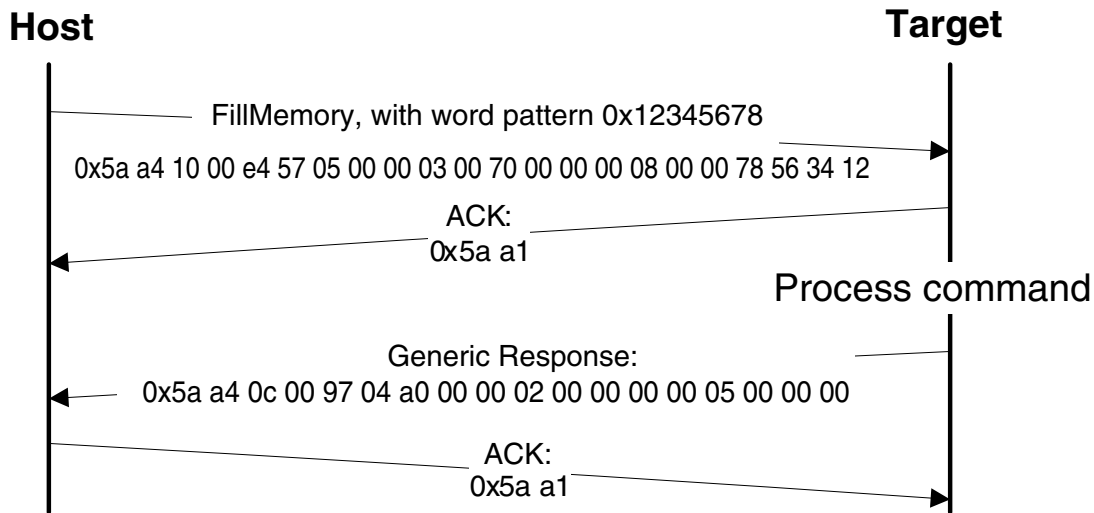
The FillMemory command fills a range of bytes in memory with a word pattern. It follows the same rules as the WriteMemory command. The difference between FillMemory and Writememory is that a word data pattern is included in FillMemory command parameter, and there is no data phase for this command, while WriteMemory does have a data phase.

**Table 13-25. Parameters for FillMemory Command**

Byte #	Command
0 - 3	Start address of memory to fill
4 - 7	Number of bytes to write with the pattern
8 - 11	Pattern word

- To fill with a byte pattern (8-bit), the byte must be replicated 4 times in the pattern word.
- To fill with a short pattern (16-bit), the short value must be replicated 2 times in the pattern word.

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.



**Figure 13-10. Protocol Sequence for FillMemory Command**

**Table 13-26. 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.

### 13.3.7.6 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.

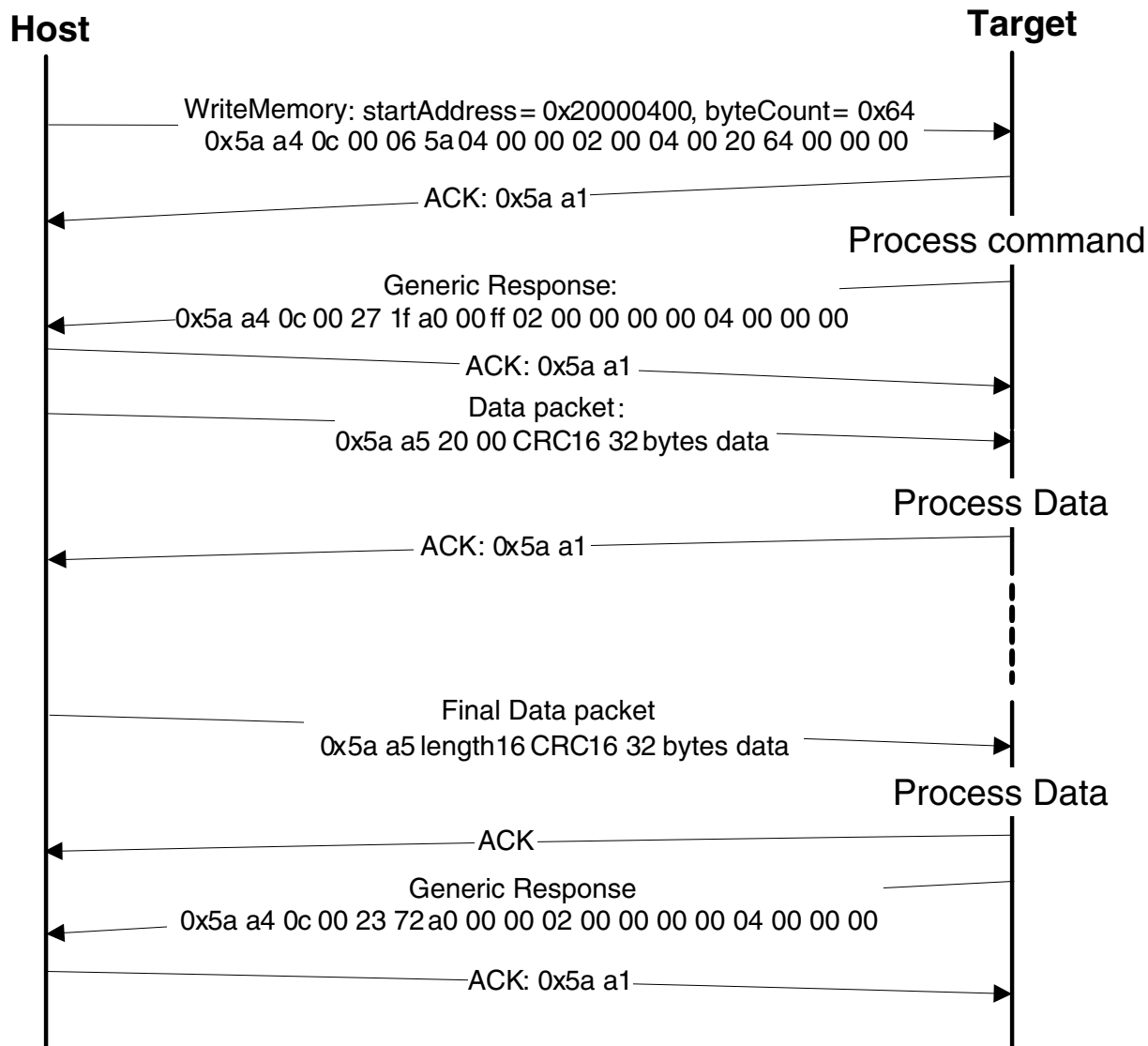
- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash requires the start address to be .
- The byte count will be rounded up to a multiple of , and the trailing bytes will be filled with the flash erase pattern (0xff).
- 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 13-27. Parameters for WriteMemory Command**

Byte #	Command
0 - 3	Start address
4 - 7	Byte count



**Figure 13-11. Protocol Sequence for WriteMemory Command**

**Table 13-28. 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.

### 13.3.7.7 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 13-29. 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

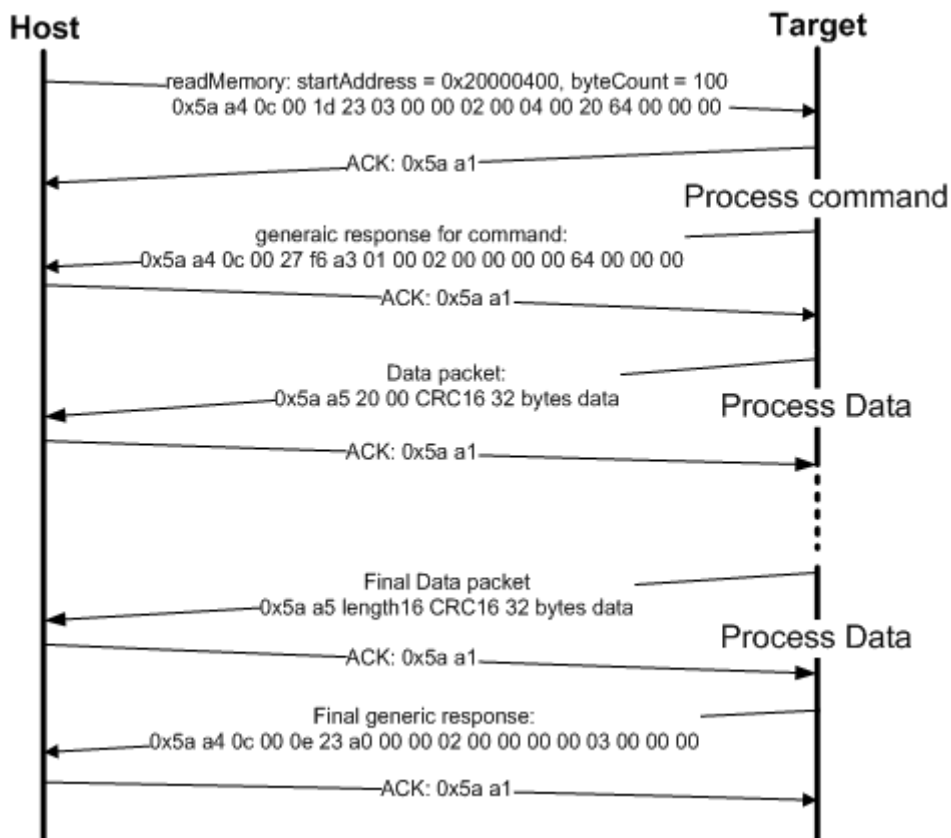


Figure 13-12. Command sequence for read memory

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 (Kinetic 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.

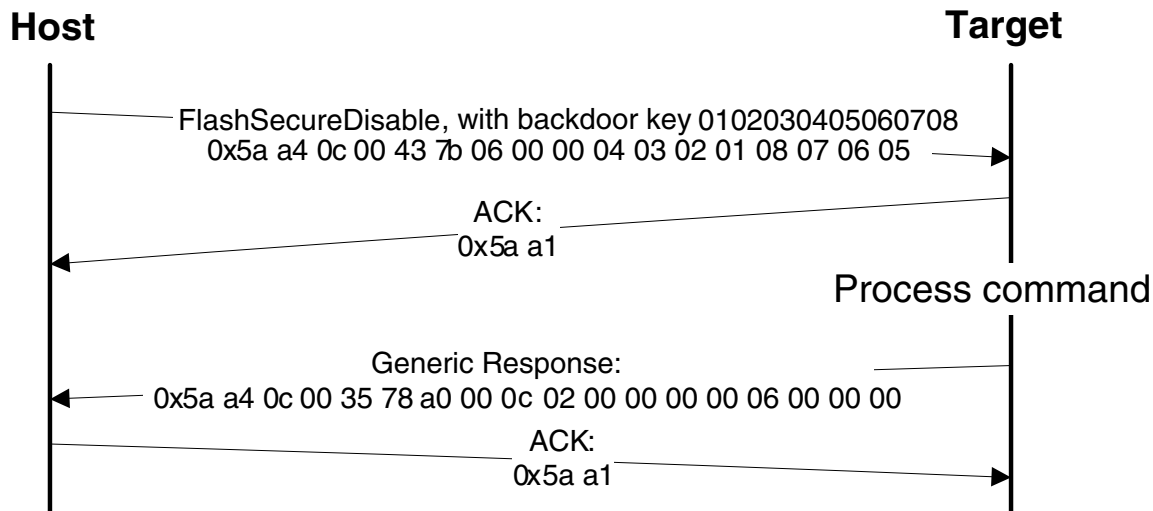
### 13.3.7.8 FlashSecurityDisable command

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 13-31. Parameters for FlashSecurityDisable Command**

Byte #	Command
0 - 3	Backdoor key low word
4 - 7	Backdoor key high word



**Figure 13-13. Protocol Sequence for FlashSecurityDisable Command**

**Table 13-32. FlashSecurityDisable Command Packet Format (Example)**

FlashSecurityDisable	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0x43 0x7B

*Table continues on the next page...*

**Table 13-32. FlashSecurityDisable Command Packet Format (Example) (continued)**

FlashSecurityDisable	Parameter	Value
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.

### 13.3.7.9 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. See the [Flashloader Exit state](#) section.

The Jump address, function argument pointer and stack pointer are the parameters required for the Execute command.

**Table 13-33. 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:** 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.

### 13.3.7.10 Reset command

The Reset command will result in flashloader resetting the chip.

The Reset command requires no parameters.

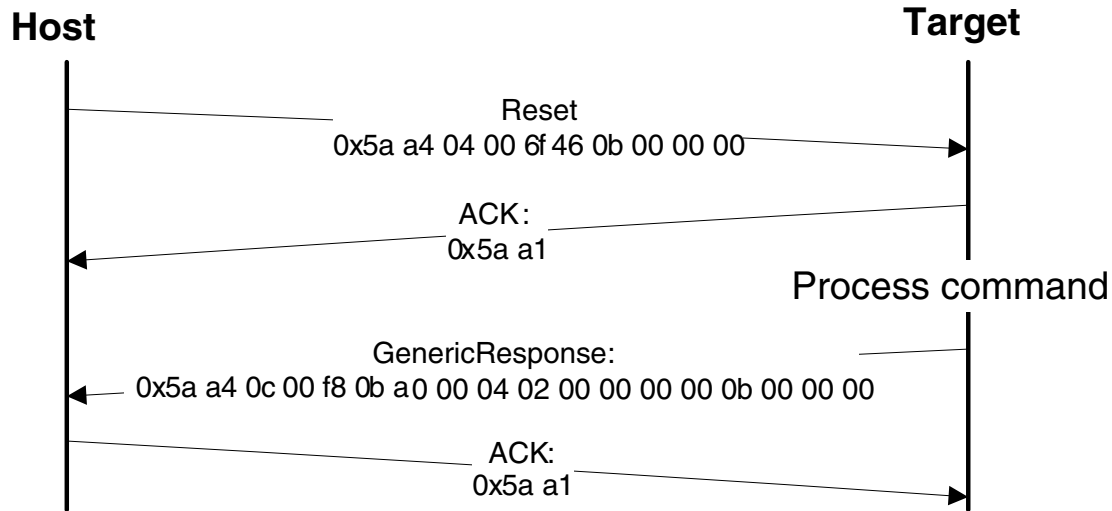


Figure 13-14. Protocol Sequence for Reset Command

Table 13-34. 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 a status code either set to kStatus\_Success upon successful execution of the command, or set to an appropriate error status code.

### 13.3.8 Flashloader Exit state

The Kinetis Flashloader tries to reconfigure the system back to the reset state in the following situations:

- After completion of an Execute command, but before jumping to the specified entry point.
- After a peripheral detection timeout, but before jumping to the application entry point.

In general, all peripherals are reset. However, the application must consider the following exceptions:

- I2C and SPI are restored by writing the control registers to reset values.
- LPUART is not restored.
- Peripheral pin mux is not reset to default, see the pin mux table ([Table 13-1](#), Kinetis Flashloader Peripheral Pinmux).
- Upon exit from the flashloader, the flashloader leaves global interrupts disabled and restores the VTOR register to its default value (0x0).

#### NOTE

PORT clock gate, pin mux and peripheral registers are not reset to default values on Flashloader exit.

- Affected PORT clock gates: PORTA, PORTB, PORTC, PORTD and PORTE (SIM\_SCGC5\_PORTA, SIM\_SCGC5\_PORTB, SIM\_SCGC5\_PORTC, SIM\_SCGC5\_PORTD and SIM\_SCGC5\_PORTE are enabled)
- Affected pin mux:
  - UART1(PTE0, PTE1)
  - I2C1(PTC10, PTC11)
  - SPI1(PTD4, PTD5, PTD6, PTD7)
- Affected peripheral registers:
  - UART and UART clock source (SIM\_SOPT2\_PLLFLLSEL = 3)
  - SPI
  - I2C

You must re-configure the corresponding register to the expected value, instead of relying on the default value.

## 13.4 Peripherals Supported

This section describes the peripherals (UART, I2C, SPI) supported by the Kinetis Flashloader.

### 13.4.1 I2C Peripheral

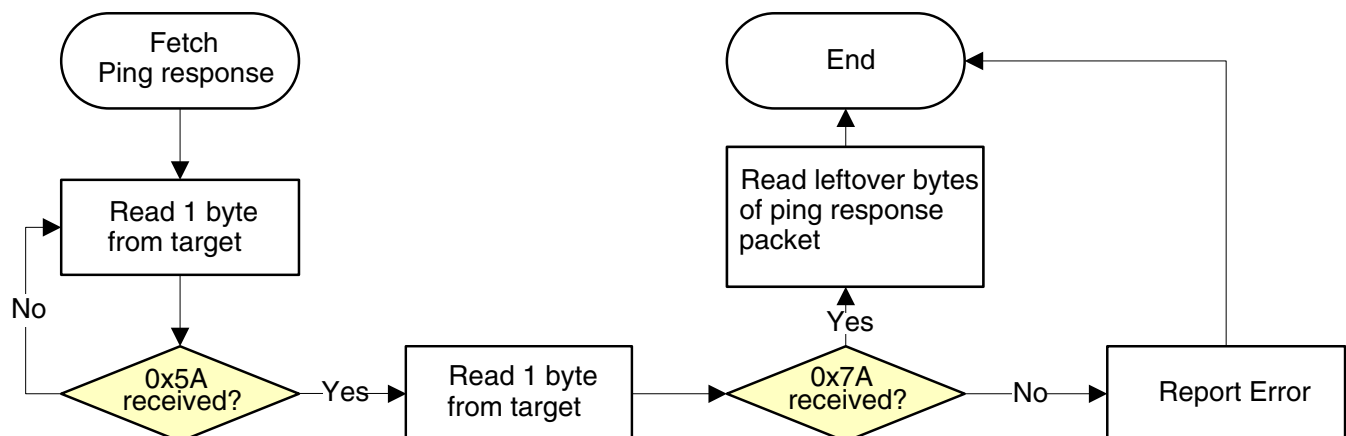
The Kinetis Flashloader supports loading data into flash via the I2C peripheral, where the I2C peripheral serves as the I2C slave. A 7-bit slave address is used during the transfer.

The Kinetis Flashloader uses 0x10 as the I2C slave address, and supports 400 kbps as the I2C baud rate.

Because the I2C peripheral in 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 13-15. Host reads ping response from target via I2C**

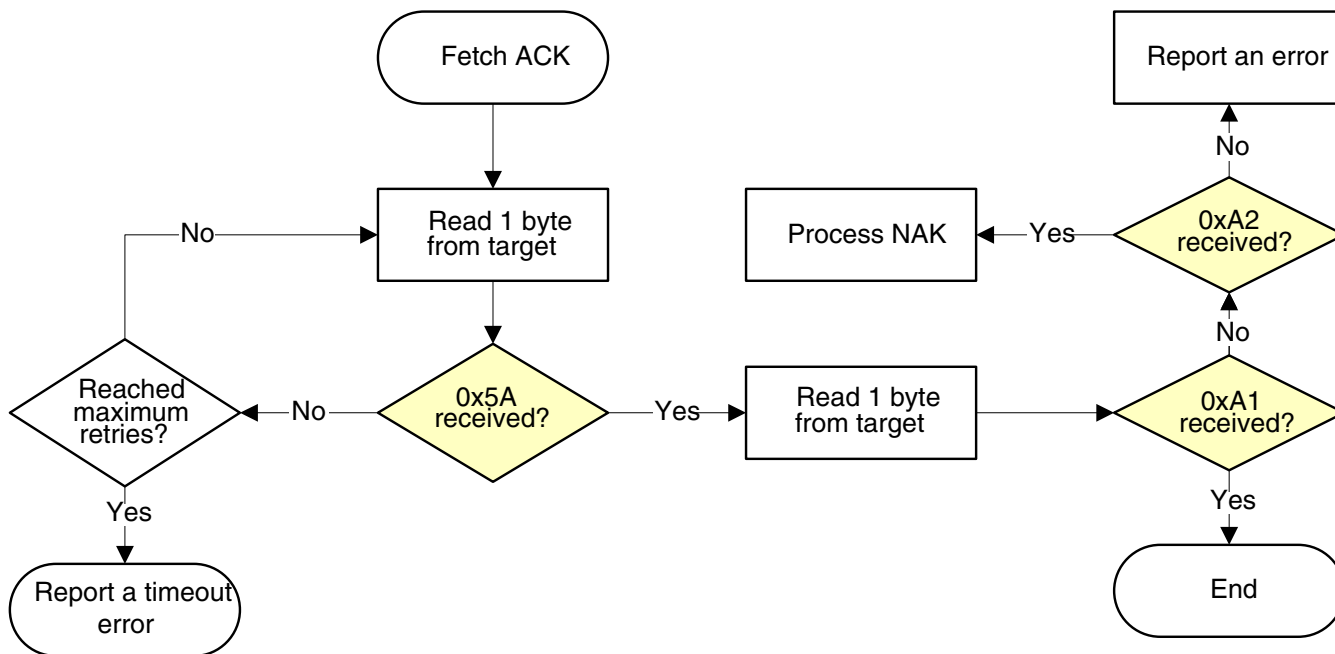


Figure 13-16. Host reads ACK packet from target via I2C

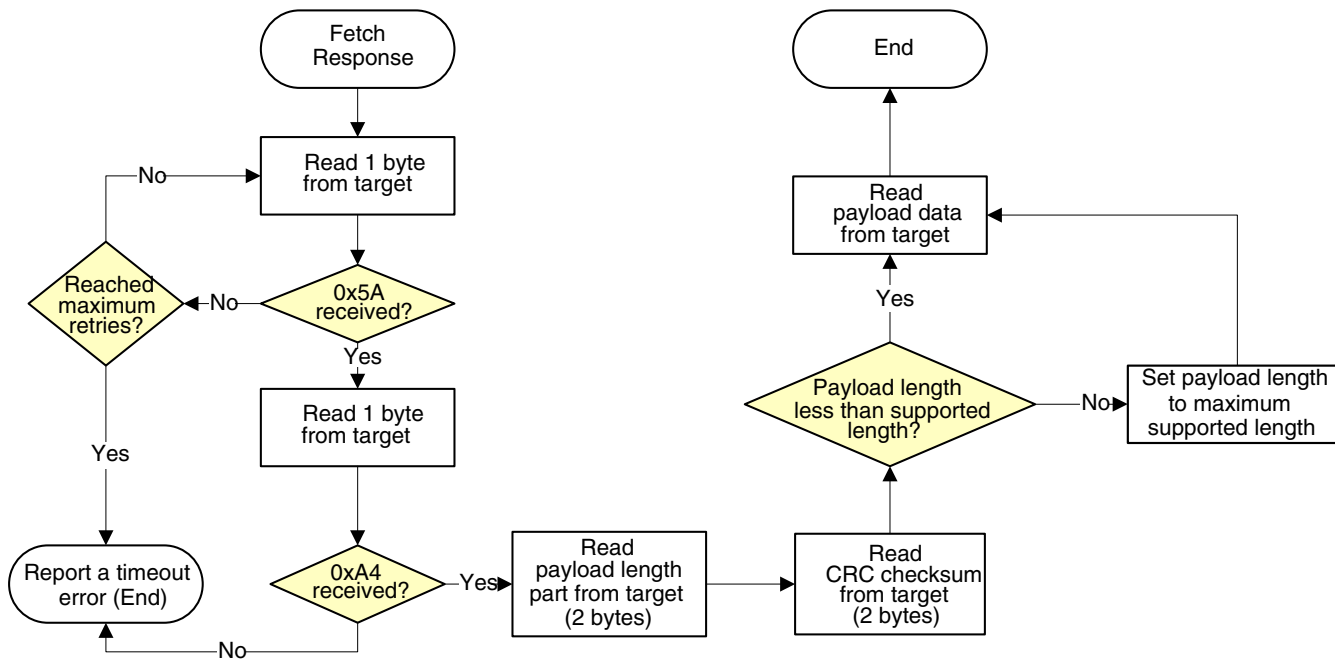


Figure 13-17. Host reads response from target via I2C

### 13.4.2 SPI Peripheral

The Kinetis Flashloader supports loading data into flash via the SPI peripheral, where the SPI peripheral serves as a SPI slave.

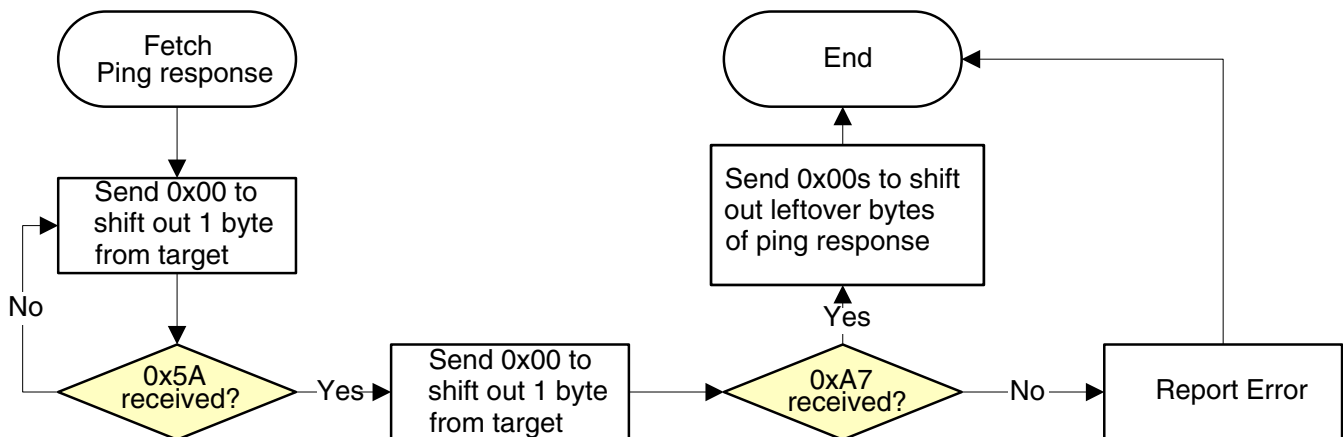
The Kinetis Flashloader supports 400 kbps as the SPI baud rate.

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 following flowcharts demonstrate how the host reads a ping response, an ACK and a command response from target via SPI.



**Figure 13-18. Host reads ping packet from target via SPI**

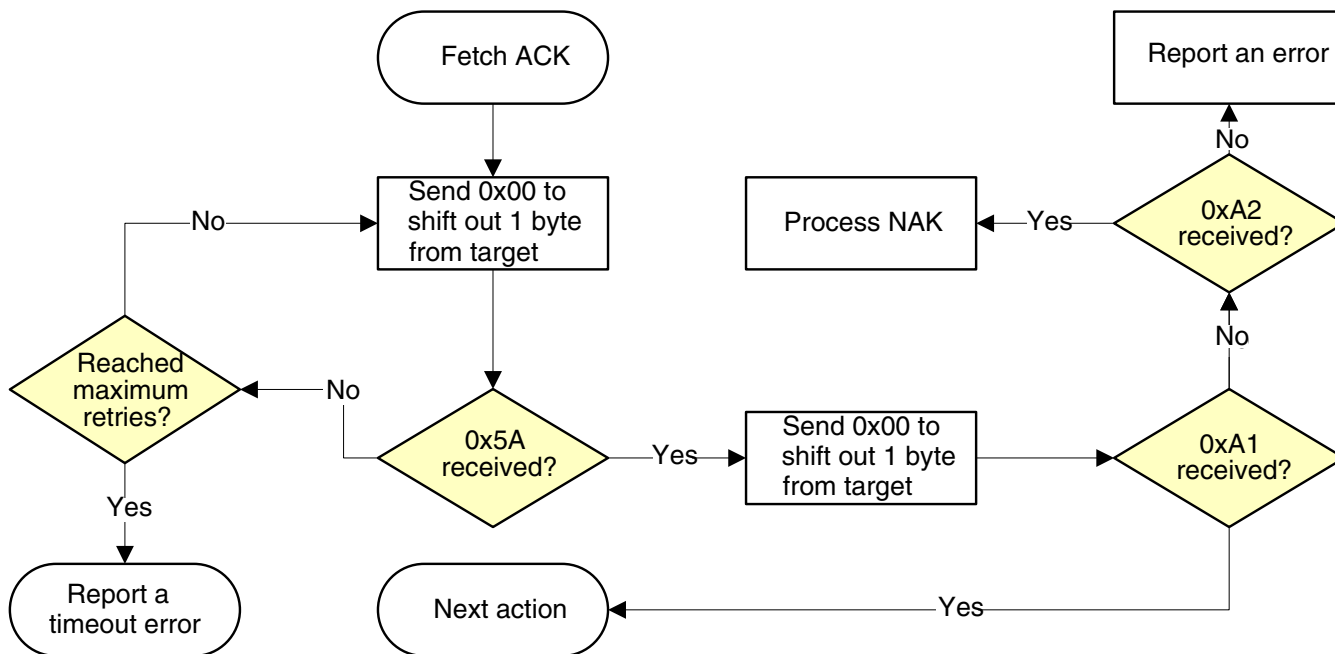


Figure 13-19. Host reads ACK from target via SPI

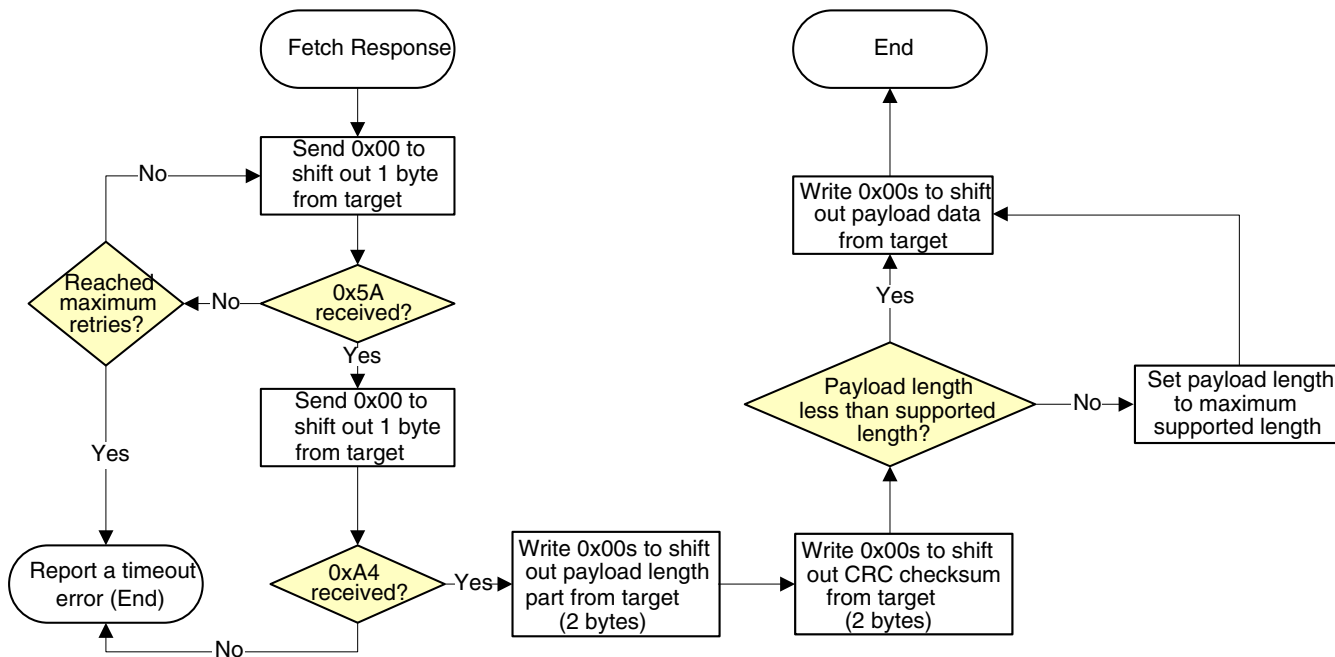


Figure 13-20. Host reads response from target via SPI

### 13.4.3 UART Peripheral

The Kinetis Flashloader integrates an autobaud detection algorithm for the UART peripheral, thereby providing flexible baud rate choices.



**Autobaud feature:** If UART1 is used to connect to the flashloader, then the UART1\_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 UART1\_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 data bytes of ping packet must be sent continuously (with no delay 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 some delay between them, 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, and 57600.

**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

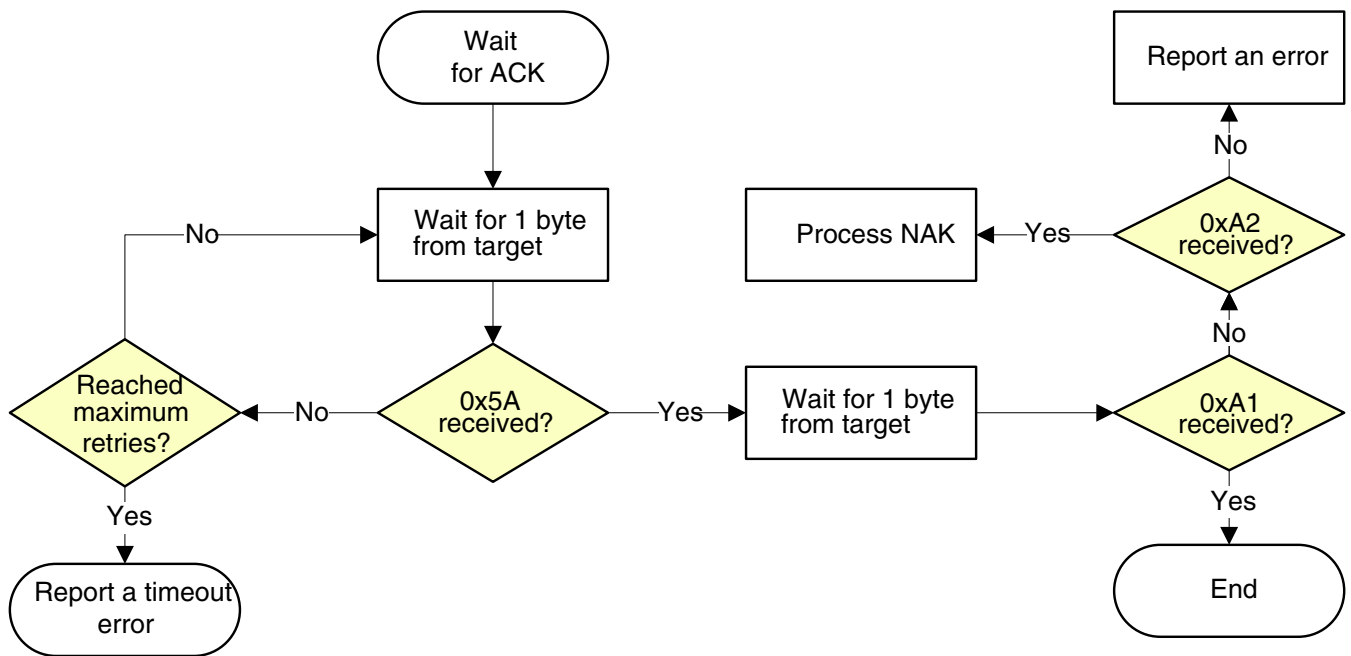


Figure 13-21. Host reads an ACK from target via UART

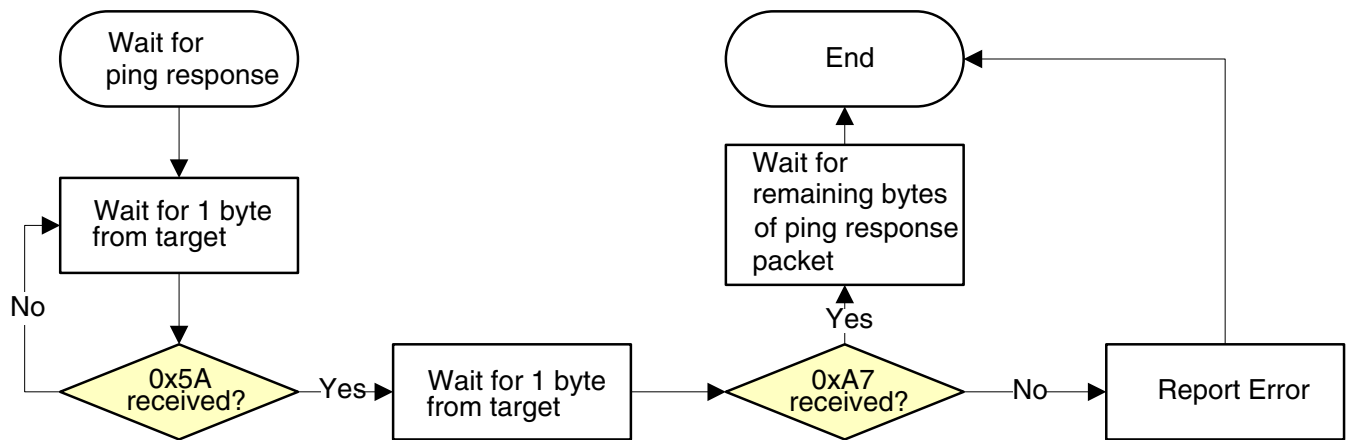


Figure 13-22. Host reads a ping response from target via UART

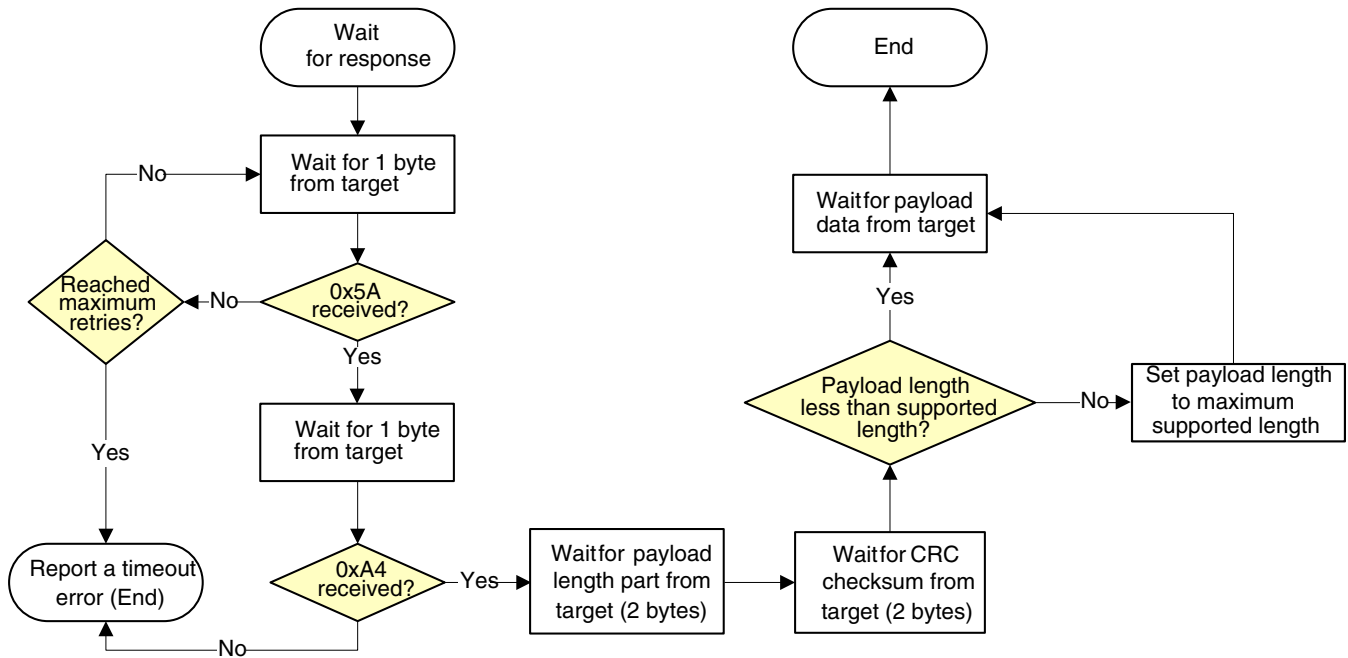


Figure 13-23. Host reads a command response from target via UART

## 13.5 Get/SetProperty Command Properties

Table 13-35. Properties used by Get/SetProperty Commands, sorted by Value

Property	Writable	Tag Value	Size	Description
<a href="#">CurrentVersion</a>	No	01h	4	Current flashloader version.
<a href="#">AvailablePeripherals</a>	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.
<a href="#">AvailableCommands</a>	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 13-35. 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>). <ul style="list-style-type: none"> <li>If HasDataPhase flag is not set, then the Response packet parameter count indicates the number of pairs.</li> <li>If HasDataPhase flag is set, then the second parameter is the number of bytes in the data phase.</li> </ul>
ValidateRegions	Yes	0Dh	4	Controls whether the flashloader will validate attempts to write to memory regions (i.e., check if they are reserved before attempting to write). ValidateRegions feature is enabled by default. 0 - No validation is done 1 - Enable validation
RAMStartAddress	No	0Eh	4	Start address of RAM
RAMSizeInBytes	No	0Fh	4	Size in bytes of RAM
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

## 13.5.1 Property Definitions

Get/Set property definitions are provided in this section.

### 13.5.1.1 CurrentVersion Property

The value of this property is a 4-byte structure containing the current version of the flashloader.

**Table 13-36. Bit ranges for the version components:**

Bits	[31:24]	[23:16]	[15:8]	[7:0]
Field	Name = 'K' (0x4B)	Major version	Minor version	Bugfix version

The Kinetis Codebase Version number returned for the flashloader is 1.1.0.

### 13.5.1.2 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 13-37. Peripheral bits:**

Bit	[31:7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Peripheral	Reserved	Reserved	Reserved	USB HID	Reserved	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.

### 13.5.1.3 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 13-38. Command bits:**

Bit	[31:11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Command	Reserved	Reserved	Call	Execute	ReceiveSBFile	GetProperty	FlashSecurityDisable	FillMemory	WriteMemory	ReadMemory	FlashEraseRegion	FlashEraseAll

## 13.6 Kinetis Flashloader Status Error Codes

This section describes the status error codes that the Kinetis Flashloader returns to the host.

**Table 13-39. 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_ I2C_SlaveTxUnderrun	200	I2C Slave TX Underrun error.
kStatus_ I2C_SlaveRxOverrun	201	I2C Slave RX Overrun error.
kStatus_ I2C_ArbitrationLost	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 transfer timed out.
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.

# Chapter 14

## Reset Control Module (RCM)

### 14.1 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 and ColdFire+ MCUs](#) for further details on using the RCM.

### 14.2 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	<a href="#">14.2.1/280</a>
4007_F001	System Reset Status Register 1 (RCM_SRS1)	8	R	00h	<a href="#">14.2.2/281</a>
4007_F004	Reset Pin Filter Control register (RCM_RPFC)	8	R/W	00h	<a href="#">14.2.3/283</a>
4007_F005	Reset Pin Filter Width register (RCM_RPFW)	8	R/W	00h	<a href="#">14.2.4/284</a>

*Table continues on the next page...*

**RCM memory map (continued)**

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4007_F008	Sticky System Reset Status Register 0 (RCM_SSRS0)	8	R/W	82h	<a href="#">14.2.5/285</a>
4007_F009	Sticky System Reset Status Register 1 (RCM_SSRS1)	8	R/W	00h	<a href="#">14.2.6/286</a>

**14.2.1 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	<p>Power-On Reset</p> <p>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.</p> <p>0 Reset not caused by POR 1 Reset caused by POR</p>
6 PIN	<p>External Reset Pin</p> <p>Indicates a reset has been caused by an active-low level on the external <math>\overline{\text{RESET}}</math> pin.</p> <p>0 Reset not caused by external reset pin 1 Reset caused by external reset pin</p>
5 WDOG	<p>Watchdog</p>

*Table continues on the next page...*



**RCM\_SRS0 field descriptions (continued)**

Field	Description
	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.
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. In LLS mode, the RESET pin is the only wakeup source that can cause this reset. 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

**14.2.2 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

## Reset memory map and register descriptions

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	JTAG
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 JTAG	JTAG Generated Reset  Indicates a reset has been caused by JTAG selection of certain IR codes: EXTEST, HIGHZ, and CLAMP.  0 Reset not caused by JTAG 1 Reset caused by JTAG

### 14.2.3 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

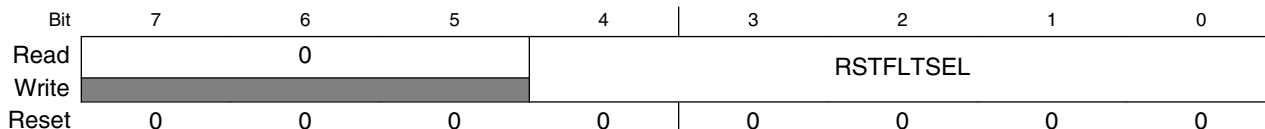
Field	Description
7-3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 RSTFLTSS	Reset Pin Filter Select in Stop Mode  Selects how the reset pin filter is enabled in Stop and VLPS modes , and also during LLS and VLLS modes. On exit from VLLS mode, this bit should be reconfigured before clearing PMC_REGSC[ACKISO].  0 All filtering disabled 1 LPO clock filter enabled
1-0 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

### 14.2.4 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



**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.
4-0 RSTFLTSEL	Reset Pin Filter Bus Clock Select Selects the reset pin bus clock filter width.  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

Table continues on the next page...

## RCM\_RPFW field descriptions (continued)

Field	Description
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

## 14.2.5 Sticky System Reset Status Register 0 (RCM\_SSRS0)

This register includes status flags to indicate all reset sources since the last POR, LVD or VLLS Wakeup that have not been cleared by software. Software can clear the status flags by writing a logic one to a flag.

Address: 4007\_F000h base + 8h offset = 4007\_F008h

Bit	7	6	5	4	3	2	1	0
Read	SPOR	SPIN	SWDOG	0		SLOC	SLVD	SWAKEUP
Write	w1c	w1c	w1c			w1c	w1c	w1c
Reset	1	0	0	0	0	0	1	0

## RCM\_SSRS0 field descriptions

Field	Description
7 SPOR	<p>Sticky Power-On Reset</p> <p>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.</p> <p>0 Reset not caused by POR 1 Reset caused by POR</p>
6 SPIN	<p>Sticky External Reset Pin</p> <p>Indicates a reset has been caused by an active-low level on the external <math>\overline{\text{RESET}}</math> pin.</p> <p>0 Reset not caused by external reset pin 1 Reset caused by external reset pin</p>
5 SWDOG	<p>Sticky Watchdog</p> <p>Indicates a reset has been caused by the watchdog timer timing out. This reset source can be blocked by disabling the watchdog.</p>

Table continues on the next page...

**RCM\_SSRS0 field descriptions (continued)**

Field	Description
	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.
2 SLOC	Sticky 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 SLVD	Sticky 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 SWAKEUP	Sticky Low Leakage Wakeup Reset  Indicates a reset has been caused by an enabled LLWU modulewakeup source while the chip was in a low leakage mode. In LLS mode, the <u>RESET</u> pin is the only wakeup source that can cause this reset. Any enabled wakeup source in a VLLSx mode causes a reset.  0 Reset not caused by LLWU module wakeup source 1 Reset caused by LLWU module wakeup source

**14.2.6 Sticky System Reset Status Register 1 (RCM\_SSRS1)**

This register includes status flags to indicate all reset sources since the last POR, LVD or VLLS Wakeup that have not been cleared by software. Software can clear the status flags by writing a logic one to a flag.

Address: 4007\_F000h base + 9h offset = 4007\_F009h

Bit	7	6	5	4	3	2	1	0
Read	0	0	SSACKERR	0	SMDM_AP	SSW	SLOCKUP	SJTAG
Write			w1c		w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

## RCM\_SSRS1 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 SSACKERR	Sticky 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 SMDM_AP	Sticky 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 SSW	Sticky 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 SLOCKUP	Sticky 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 SJTAG	Sticky JTAG Generated Reset  Indicates a reset has been caused by JTAG selection of certain IR codes: EXTEST, HIGHZ, and CLAMP.  0 Reset not caused by JTAG 1 Reset caused by JTAG





# Chapter 15

## System Mode Controller (SMC)

### 15.1 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 and ColdFire+ MCUs](#) for further details on using the SMC.

### 15.2 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 Freescale microcontrollers.

## Modes of operation

The following table shows the translation between the ARM CPU modes and the Freescale 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 Freescale MCU documentation normally uses wait and stop.

In addition, Freescale 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 15-1. 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.
HSRUN	The MCU can be run at a faster frequency compared with RUN mode and the internal supply is fully regulated. See the Power Management chapter for details about the maximum allowable frequencies.
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.

*Table continues on the next page...*

**Table 15-1. Power modes (continued)**

Mode	Description
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.
LLS3	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 reducing the voltage to internal logic. All system RAM contents, internal logic and I/O states are retained.
LLS2	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 reducing voltage to internal logic and powering down the system RAM2 partition. The system RAM1 partition, internal logic and I/O states are retained. <sup>1</sup>
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.
VLLS2	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 the system RAM2 partition. The system RAM1 partition contents are retained in this mode. Internal logic states are not retained. <sup>2</sup>
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].

1. See the devices' chip configuration details for the size and location of the system RAM partitions.
2. See the devices' chip configuration details for the size and location of the system RAM partitions.

## 15.3 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	<a href="#">15.3.1/292</a>
4007_E001	Power Mode Control register (SMC_PMCTRL)	8	R/W	00h	<a href="#">15.3.2/293</a>
4007_E002	Stop Control Register (SMC_STOPCTRL)	8	R/W	03h	<a href="#">15.3.3/295</a>
4007_E003	Power Mode Status register (SMC_PMSTAT)	8	R	01h	<a href="#">15.3.4/296</a>

**15.3.1 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	AHSTRUN	0	AVLP	0	ALLS	0	AVLLS	0
Write								
Reset	0	0	0	0	0	0	0	0

**SMC\_PMPROT field descriptions**

Field	Description
7 AHSRUN	<p>Allow High Speed Run mode</p> <p>Provided the appropriate control bits are set up in PMCTRL, this write-once field allows the MCU to enter High Speed Run mode (HSRUN).</p> <p>0 HSRUN is not allowed 1 HSRUN is allowed</p>
6 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
5 AVLP	<p>Allow Very-Low-Power Modes</p> <p>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).</p> <p>0 VLPR, VLPW, and VLPS are not allowed. 1 VLPR, VLPW, and VLPS are allowed.</p>
4 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
3 ALLS	<p>Allow Low-Leakage Stop Mode</p> <p>Provided the appropriate control bits are set up in PMCTRL, this write-once field allows the MCU to enter any low-leakage stop mode (LLS).</p> <p>0 Any LLSx mode is not allowed 1 Any LLSx mode is allowed</p>
2 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
1 AVLLS	<p>Allow Very-Low-Leakage Stop Mode</p> <p>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).</p> <p>0 Any VLLSx mode is not allowed 1 Any VLLSx mode is allowed</p>
0 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>

**15.3.2 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.

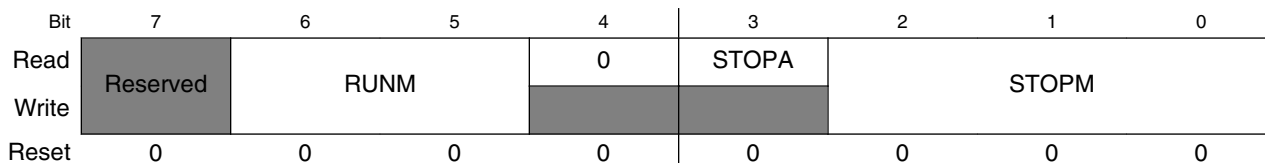
**NOTE**

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

**Memory map and register descriptions**

that do not trigger Chip POR not VLLS. See the Reset section details for more information.

Address: 4007\_E000h base + 1h offset = 4007\_E001h



**SMC\_PMCTRL field descriptions**

Field	Description
7 Reserved	This field is reserved. This bit is reserved for future expansion and should always be written zero.
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.  <b>NOTE:</b> 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.  <b>NOTE:</b> RUNM may be set to HSRUN only when PMSTAT=RUN. After being programmed to HSRUN, RUNM should not be programmed back to RUN until PMSTAT=HSRUN. Also, stop mode entry should not be attempted while RUNM=HSRUN or PMSTAT=HSRUN.  00 Normal Run mode (RUN) 01 Reserved 10 Very-Low-Power Run mode (VLPR) 11 High Speed Run mode (HSRUN)
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 occurred during the previous stop mode entry sequence, preventing the system from entering that mode. This field is cleared by reset or 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.
2–0 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.  <b>NOTE:</b> When set to VLLSx or LLSx, the LLSM in the STOPCTRL register is used to further select the particular VLLS or LLS submode which will be entered.  <b>NOTE:</b> 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)

Table continues on the next page...

**SMC\_PMCTRL field descriptions (continued)**

Field	Description
001	Reserved
010	Very-Low-Power Stop (VLPS)
011	Low-Leakage Stop (LLSx)
100	Very-Low-Leakage Stop (VLLSx)
101	Reserved
110	Reserved
111	Reserved

**15.3.3 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	LLSM	
Write								
Reset	0	0	0	0	0	0	1	1

**SMC\_STOPCTRL field descriptions**

Field	Description
7–6 PSTOPO	<p>Partial Stop Option</p> <p>These bits control whether a Partial Stop mode is entered when STOPM=STOP. When entering a Partial Stop mode from RUN 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.</p> <p>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</p>
5 PORPO	<p>POR Power Option</p> <p>This bit controls whether the POR detect circuit is enabled in VLLS0 mode.</p>

*Table continues on the next page...*

**SMC\_STOPCTRL field descriptions (continued)**

Field	Description
	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 and should always be written zero.
2-0 LLSM	LLS or VLLS Mode Control This field controls which LLS or VLLS sub-mode to enter if STOPM = LLSx or VLLSx.  000 VLLS0 if PMCTRL[STOPM]=VLLSx, reserved if PMCTRL[STOPM]=LLSx 001 VLLS1 if PMCTRL[STOPM]=VLLSx, reserved if PMCTRL[STOPM]=LLSx 010 VLLS2 if PMCTRL[STOPM]=VLLSx, LLS2 if PMCTRL[STOPM]=LLSx 011 VLLS3 if PMCTRL[STOPM]=VLLSx, LLS3 if PMCTRL[STOPM]=LLSx 100 Reserved 101 Reserved 110 Reserved 111 Reserved

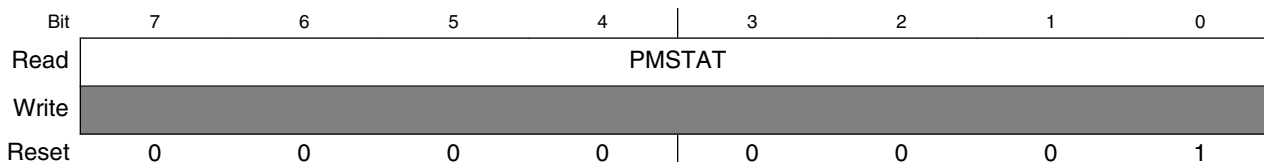
**15.3.4 Power Mode Status register (SMC\_PMSTAT)**

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



**SMC\_PMSTAT field descriptions**

Field	Description
7-0 PMSTAT	Power Mode Status  <b>NOTE:</b> When debug is enabled, the PMSTAT will not update to STOP or VLPS



**SMC\_PMSTAT field descriptions (continued)**

Field	Description
	<p><b>NOTE:</b> When a PSTOP mode is enabled, the PMSTAT will not update to STOP or VLPS</p> <p>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 Current power mode is LLS.            0100_0000 Current power mode is VLLS.            1000_0000 Current power mode is HSRUN</p>

## 15.4 Functional description

### 15.4.1 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.

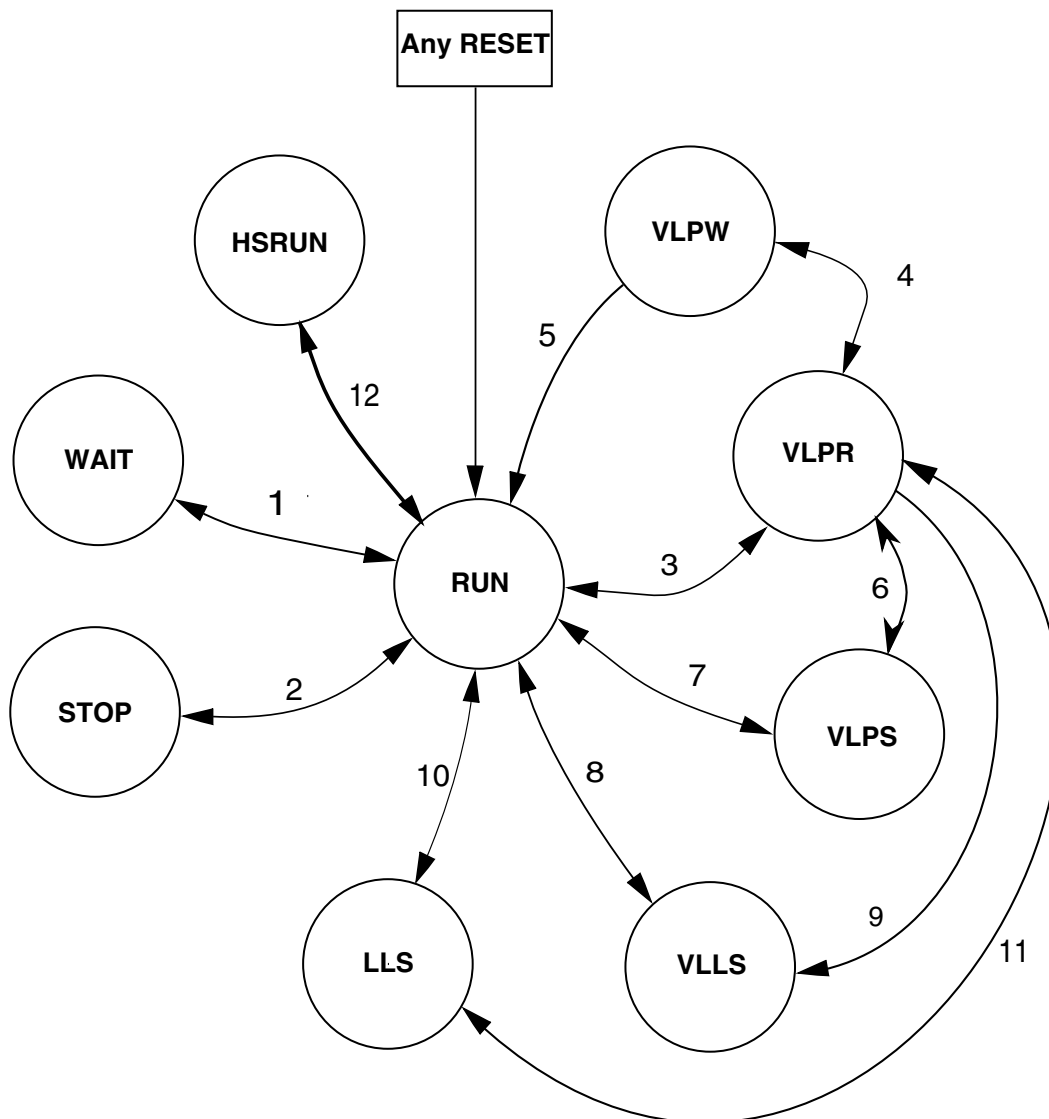


Figure 15-5. Power mode state diagram

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

Table 15-7. 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. <sup>1</sup>
	WAIT	RUN	Interrupt or Reset

Table continues on the next page...

Table 15-7. Power mode transition triggers (continued)

Transition #	From	To	Trigger conditions
2	RUN	STOP	PMCTRL[RUNM]=00, PMCTRL[STOPM]=000 <sup>2</sup> Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core. See note. <sup>1</sup>
	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. <sup>1</sup>
	VLPW	VLPR	Interrupt
5	VLPW	RUN	Reset
6	VLPR	VLPS	PMCTRL[STOPM]=000 <sup>3</sup> 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. <sup>1</sup>
	VLPS	VLPR	Interrupt <b>NOTE:</b> If VLPS was entered directly from RUN (transition #7), 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. <sup>1</sup>
	VLPS	RUN	Interrupt and VLPS mode was entered directly from RUN or Reset
8	RUN	VLLSx	PMPROT[AVLLS]=1, PMCTRL[STOPM]=100, STOPCTRL[VLLSM]=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[VLLSM]=x (VLLSx), Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.

Table continues on the next page...

**Table 15-7. Power mode transition triggers (continued)**

Transition #	From	To	Trigger conditions
10	RUN	LLS	PMPROT[ALLS]=1, PMCTRL[STOPM]=011, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.
	LLS	RUN	Wakeup from enabled LLWU input source and LLS mode was entered directly from RUN or RESET pin.
11	VLPR	LLS	PMPROT[ALLS]=1, PMCTRL[STOPM]=011, Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core.
	LLS	VLPR	Wakeup from enabled LLWU input source and LLS mode was entered directly from VLPR <b>NOTE:</b> If LLS was entered directly from RUN, hardware will not allow this transition and will force exit back to RUN
12	RUN	HSRUN	Set PMPROT[AHSRUN]=1, PMCTRL[RUNM]=11.
	HSRUN	RUN	Set PMCTRL[RUNM]=00 or Reset

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

## 15.4.2 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 15-6. Low-power system components and connections**

### 15.4.2.1 Stop mode entry sequence

Entry into a low-power stop mode (Stop, VLPS, LLS, VLLSx) is initiated by CPU execution of 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.

### 15.4.2.2 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.

### 15.4.2.3 Aborted stop mode entry

If an interrupt 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 interrupt occurs before the PMC begins the transition to stop mode regulation. After this point, the interrupt 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.

### 15.4.2.4 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.

### 15.4.2.5 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.

### 15.4.3 Run modes

The run modes supported by this device can be found here.

- Run (RUN)
- Very Low-Power Run (VLPR)
- High Speed Run (HSRUN)

#### 15.4.3.1 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 using their corresponding clock gating control bits in the SIM's registers.

#### 15.4.3.2 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] is 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.

### 15.4.3.3 High Speed Run (HSRUN) mode

In HSRUN mode, the on-chip voltage regulator remains in a run regulation state, but with a slightly elevated voltage output. In this state, the MCU is able to operate at a faster frequency compared to normal RUN mode. See Power Management chapter for maximum allowable frequencies.

While in this mode, the following restrictions must be adhered to:

- The maximum allowable change in frequency of the system, bus, flash or core clocks is restricted to x2.
- Before exiting HSRUN mode, clock frequencies should be reduced back down to those acceptable in RUN mode.
- Stop mode entry is not supported from HSRUN.
- Modifications to clock gating control bits are prohibited.
- Flash programming/erasing is not allowed.

To enter HSRUN mode, set RUNM=HSRUN. Before increasing clock frequencies, the PMSTAT register should be polled to determine when the system has completed entry into HSRUN mode. To reenter normal RUN mode, clear RUNM. Any reset will also clear RUNM and cause the system to exit to normal RUN mode after the MCU exits its reset flow.

## 15.4.4 Wait modes

This device contains two different wait modes which are listed here.

- Wait
- Very-Low Power Wait (VLPW)

### 15.4.4.1 WAIT mode

WAIT mode is entered when the ARM core enters the Sleep-Now or Sleep-On-Exit modes while SLEEDEEP 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. Clock gating to the peripheral is enabled via the SIM module.

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 will cause an exit from WAIT mode, returning the device to normal RUN mode.

### 15.4.4.2 Very-Low-Power Wait (VLPW) mode

VLPW is entered by the entering the Sleep-Now or Sleep-On-Exit mode while SLEEPDEEP is cleared and the MCU 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 MCU over a reduced frequency. To further reduce power in this mode, disable the clocks to unused modules by clearing the peripherals' corresponding clock gating control bits in the SIM.

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 will cause an exit from VLPW mode, returning the device to normal RUN mode.

## 15.4.5 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.

### NOTE

All clock monitors must be disabled before entering these low-power modes: Stop, VLPS, VLPR, VLPW, LLS, and VLLSx.

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)
- Low-Leakage Stop (LLS)
- Very-Low-Leakage Stop (VLLSx)

### 15.4.5.1 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.

### 15.4.5.2 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.

### 15.4.5.3 Low-Leakage Stop (LLSx) modes

This device contains two Low-Leakage Stop modes: LLS3 and LLS2. LLS or LLSx is often used in this document to refer to both modes. All LLS modes can be entered from normal RUN or VLPR modes.

The MCU enters LLS mode if:

- In Sleep-Now or Sleep-On-Exit mode, SLEEPDEEP is set in the System Control Register in the ARM core, and
- The device is configured as shown in [Table 15-7](#).

In LLS, the on-chip voltage regulator is in stop regulation. Most of the peripherals are put in a state-retention mode that does not allow them to operate while in LLS.

Before entering LLS mode, the user should configure the Low-Leakage Wake-up (LLWU) module to enable the desired wake-up sources. The available wake-up sources in LLS are detailed in the chip configuration details for this device.

After wakeup from LLS, the device returns to the run mode from which LLS was entered (either normal RUN or VLPR) with a pending LLWU module interrupt. In the LLWU interrupt service routine (ISR), the user can poll the LLWU module wake-up flags to determine the source of the wakeup.

### NOTE

The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit Stop mode on an LLS recovery.

An asserted  $\overline{\text{RESET}}$  pin will cause an exit from LLS mode, returning the device to normal RUN mode. When LLS is exiting via the  $\overline{\text{RESET}}$  pin, RCM\_SRS0[PIN] and RCM\_SRS0[WAKEUP] are set.

#### 15.4.5.4 Very-Low-Leakage Stop (VLLSx) modes

This device contains these very low leakage modes:

- VLLS3
- VLLS2
- 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 15-7](#).

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\_SRS0[PIN] and RCM\_SRS0[WAKEUP] are set.

### 15.4.6 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 LLS or VLLS modes. LLS is a state-retention mode and all debug operation can continue after waking from LLS, even in cases where system wakeup is due to a system reset event.

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 16

## Power Management Controller (PMC)

### 16.1 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.

See [AN4503: Power Management for Kinetis and ColdFire+ MCUs](#) for further details on using the PMC.

### 16.2 Features

A list of included PMC features can be found here.

- Internal voltage regulator
- Active POR providing brown-out detect
- Low-voltage detect supporting two low-voltage trip points with four warning levels per trip point

### 16.3 Low-voltage detect (LVD) system

This device includes a system to guard against low-voltage conditions. This protects memory contents and controls MCU system states during supply voltage variations.

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 `LVDS1[LVDS]`. The LVD is disabled upon entering `VLPx`, `LLS`, 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[LWVACK], but only if the internal supply has returned above the trip point; otherwise, LVDSC2[LVWF] remains set.

### **16.3.1 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\_SRS0[LVD]) is set following an LVD or power-on reset.

### **16.3.2 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].

### **16.3.3 Low-voltage warning (LVW) interrupt operation**

The LVD system 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[LWVACK].



LVVSC2[LVVW] selects one of the four trip voltages:

- Highest:  $V_{LVW4}$
- Two mid-levels:  $V_{LVW3}$  and  $V_{LVW2}$
- Lowest:  $V_{LVW1}$

## 16.4 I/O retention

When in LLS mode, the I/O pins are held in their input or output state.

Upon wakeup, the PMC is re-enabled, goes through a power up sequence to full regulation, and releases the logic from state retention mode. The I/O are released immediately after a wake-up or reset event. In the case of LLS exit via a RESET pin, the I/O default to their reset state.

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.

## 16.5 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	<a href="#">16.5.1/314</a>

*Table continues on the next page...*

PMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4007_D001	Low Voltage Detect Status And Control 2 register (PMC_LVDSC2)	8	R/W	00h	<a href="#">16.5.2/315</a>
4007_D002	Regulator Status And Control register (PMC_REGSC)	8	R/W	04h	<a href="#">16.5.3/316</a>

### 16.5.1 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

Table continues on the next page...

## PMC\_LVDSC1 field descriptions (continued)

Field	Description
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
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.
1–0 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

### 16.5.2 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.

## Memory map and register descriptions

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	<p>Low-Voltage Warning Flag</p> <p>This read-only status field indicates a low-voltage warning event. LVWF is set when <math>V_{Supply}</math> transitions below the trip point, or after reset and <math>V_{Supply}</math> is already below <math>V_{LVW}</math>. 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.</p> <p>0 Low-voltage warning event not detected 1 Low-voltage warning event detected</p>
6 LVWACK	<p>Low-Voltage Warning Acknowledge</p> <p>This write-only field is used to acknowledge low voltage warning errors. Write 1 to clear LVWF. Reads always return 0.</p>
5 LVWIE	<p>Low-Voltage Warning Interrupt Enable</p> <p>Enables hardware interrupt requests for LVWF.</p> <p>0 Hardware interrupt disabled (use polling) 1 Request a hardware interrupt when LVWF = 1</p>
4–2 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
1–0 LVWV	<p>Low-Voltage Warning Voltage Select</p> <p>Selects the LVW trip point voltage (<math>V_{LVW}</math>). The actual voltage for the warning depends on LVDSC1[LVDV].</p> <p>00 Low trip point selected (<math>V_{LVW} = V_{LVW1}</math>) 01 Mid 1 trip point selected (<math>V_{LVW} = V_{LVW2}</math>) 10 Mid 2 trip point selected (<math>V_{LVW} = V_{LVW3}</math>) 11 High trip point selected (<math>V_{LVW} = V_{LVW4}</math>)</p>

### 16.5.3 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, LLS, 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.  <b>NOTE:</b> 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 , LLS , and VLLSx modes. 1 Bandgap voltage reference is enabled in VLPx , LLS , 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.  <b>NOTE:</b> 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.  <b>NOTE:</b> This reserved bit must remain cleared (set to 0).

Table continues on the next page...

**PMC\_REGSC field descriptions (continued)**

Field	Description
0 BGBE	Bandgap Buffer Enable Enables the bandgap buffer. 0 Bandgap buffer not enabled 1 Bandgap buffer enabled

# Chapter 17

## Low-Leakage Wakeup Unit (LLWU)

### 17.1 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 16 external pins and up to 8 internal modules as interrupt wake-up sources from low-leakage power modes.

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 both LLS and VLLS through a reset flow.

The LLWU module also includes two optional digital pin filters for the external wakeup pins.

See [AN4503: Power Management for Kinetis and ColdFire+ MCUs](#) for further details on using the LLWU.

#### 17.1.1 Features

The LLWU module features include:

- Support for up to 16 external input pins and up to 8 internal modules with individual enable bits for MCU interrupt from low leakage modes
- Input sources may be external pins or from internal peripherals capable of running in LLS or 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.

### 17.1.2 Modes of operation

The LLWU module becomes functional on entry into a low-leakage power mode. After recovery from LLS, the LLWU is immediately disabled. 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].

#### 17.1.2.1 LLS mode

Wake-up events due to external pin inputs (LLWU\_Px) and internal module interrupt inputs (LLWU\_MxIF) result in an interrupt flow when exiting LLS.

#### NOTE

The LLWU interrupt must not be masked by the interrupt controller to avoid a scenario where the system does not fully exit Stop mode on an LLS recovery.

#### 17.1.2.2 VLLS modes

All wakeup and reset events result in VLLS exit via a reset flow.

#### 17.1.2.3 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.



### 17.1.2.4 Debug mode

When the chip is in Debug mode and then enters LLS or a VLLSx mode, no debug logic works in the fully-functional low-leakage mode. Upon an exit from the LLS or VLLSx mode, the LLWU becomes inactive.

### 17.1.3 Block diagram

The following figure is the block diagram for the LLWU module.

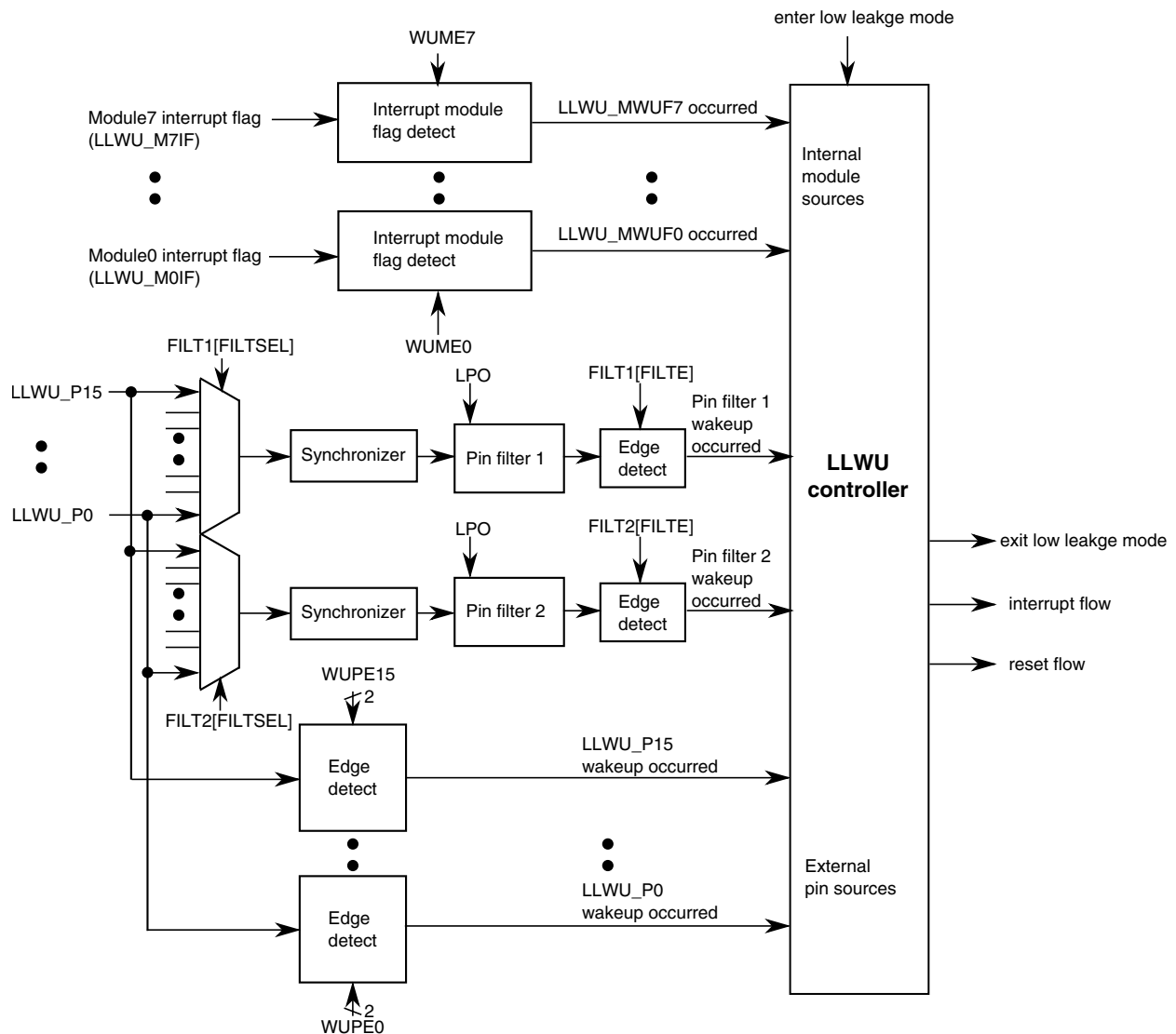


Figure 17-1. LLWU block diagram

## 17.2 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 17-1. LLWU signal descriptions**

Signal	Description	I/O
LLWU_Pn	Wakeup inputs (n = 0-15 )	I

## 17.3 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
- 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	<a href="#">17.3.1/323</a>
4007_C001	LLWU Pin Enable 2 register (LLWU_PE2)	8	R/W	00h	<a href="#">17.3.2/324</a>
4007_C002	LLWU Pin Enable 3 register (LLWU_PE3)	8	R/W	00h	<a href="#">17.3.3/325</a>
4007_C003	LLWU Pin Enable 4 register (LLWU_PE4)	8	R/W	00h	<a href="#">17.3.4/326</a>
4007_C004	LLWU Module Enable register (LLWU_ME)	8	R/W	00h	<a href="#">17.3.5/327</a>
4007_C005	LLWU Flag 1 register (LLWU_F1)	8	R/W	00h	<a href="#">17.3.6/329</a>
4007_C006	LLWU Flag 2 register (LLWU_F2)	8	R/W	00h	<a href="#">17.3.7/331</a>
4007_C007	LLWU Flag 3 register (LLWU_F3)	8	R	00h	<a href="#">17.3.8/332</a>
4007_C008	LLWU Pin Filter 1 register (LLWU_FILT1)	8	R/W	00h	<a href="#">17.3.9/334</a>
4007_C009	LLWU Pin Filter 2 register (LLWU_FILT2)	8	R/W	00h	<a href="#">17.3.10/335</a>

### 17.3.1 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	WUPE3		WUPE2		WUPE1		WUPE0	
Write								
Reset	0	0	0	0	0	0	0	0

#### LLWU\_PE1 field descriptions

Field	Description
7–6 WUPE3	<p>Wakeup Pin Enable For LLWU_P3</p> <p>Enables and configures the edge detection for the wakeup pin.</p> <p>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</p>
5–4 WUPE2	<p>Wakeup Pin Enable For LLWU_P2</p> <p>Enables and configures the edge detection for the wakeup pin.</p>

*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
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
1-0 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

**17.3.2 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	WUPE7		WUPE6		WUPE5		WUPE4	
Write								
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

*Table continues on the next page...*

**LLWU\_PE2 field descriptions (continued)**

Field	Description
	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
1–0 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

**17.3.3 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	WUPE11		WUPE10		WUPE9		WUPE8	
Write								
Reset	0	0	0	0	0	0	0	0

## LLWU\_PE3 field descriptions

Field	Description
7–6 WUPE11	<p>Wakeup Pin Enable For LLWU_P11</p> <p>Enables and configures the edge detection for the wakeup pin.</p> <p>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</p>
5–4 WUPE10	<p>Wakeup Pin Enable For LLWU_P10</p> <p>Enables and configures the edge detection for the wakeup pin.</p> <p>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</p>
3–2 WUPE9	<p>Wakeup Pin Enable For LLWU_P9</p> <p>Enables and configures the edge detection for the wakeup pin.</p> <p>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</p>
1–0 WUPE8	<p>Wakeup Pin Enable For LLWU_P8</p> <p>Enables and configures the edge detection for the wakeup pin.</p> <p>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</p>

### 17.3.4 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	WUPE15		WUPE14		WUPE13		WUPE12	
Write								
Reset	0	0	0	0	0	0	0	0

**LLWU\_PE4 field descriptions**

Field	Description
7–6 WUPE15	<p>Wakeup Pin Enable For LLWU_P15</p> <p>Enables and configures the edge detection for the wakeup pin.</p> <p>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</p>
5–4 WUPE14	<p>Wakeup Pin Enable For LLWU_P14</p> <p>Enables and configures the edge detection for the wakeup pin.</p> <p>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</p>
3–2 WUPE13	<p>Wakeup Pin Enable For LLWU_P13</p> <p>Enables and configures the edge detection for the wakeup pin.</p> <p>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</p>
1–0 WUPE12	<p>Wakeup Pin Enable For LLWU_P12</p> <p>Enables and configures the edge detection for the wakeup pin.</p> <p>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</p>

**17.3.5 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

**Memory map/register definition**

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	WUME7	WUME6	WUME5	WUME4	WUME3	WUME2	WUME1	WUME0
Write								
Reset	0	0	0	0	0	0	0	0

**LLWU\_ME field descriptions**

Field	Description
7 WUME7	<p>Wakeup Module Enable For Module 7</p> <p>Enables an internal module as a wakeup source input.</p> <p>0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source</p>
6 WUME6	<p>Wakeup Module Enable For Module 6</p> <p>Enables an internal module as a wakeup source input.</p> <p>0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source</p>
5 WUME5	<p>Wakeup Module Enable For Module 5</p> <p>Enables an internal module as a wakeup source input.</p> <p>0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source</p>
4 WUME4	<p>Wakeup Module Enable For Module 4</p> <p>Enables an internal module as a wakeup source input.</p> <p>0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source</p>
3 WUME3	<p>Wakeup Module Enable For Module 3</p> <p>Enables an internal module as a wakeup source input.</p> <p>0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source</p>
2 WUME2	<p>Wakeup Module Enable For Module 2</p> <p>Enables an internal module as a wakeup source input.</p> <p>0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source</p>
1 WUME1	<p>Wakeup Module Enable for Module 1</p> <p>Enables an internal module as a wakeup source input.</p> <p>0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source</p>

*Table continues on the next page...*



**LLWU\_ME field descriptions (continued)**

Field	Description
0 WUME0	<p>Wakeup Module Enable For Module 0</p> <p>Enables an internal module as a wakeup source input.</p> <p>0 Internal module flag not used as wakeup source 1 Internal module flag used as wakeup source</p>

**17.3.6 LLWU Flag 1 register (LLWU\_F1)**

LLWU\_F1 contains the wakeup flags indicating which wakeup source caused the MCU to exit LLS or VLLS mode. For LLS, this is the source causing the CPU interrupt flow. 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 + 5h offset = 4007\_C005h

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\_F1 field descriptions**

Field	Description
7 WUF7	<p>Wakeup Flag For LLWU_P7</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF7.</p> <p>0 LLWU_P7 input was not a wakeup source 1 LLWU_P7 input was a wakeup source</p>
6 WUF6	<p>Wakeup Flag For LLWU_P6</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF6.</p>

*Table continues on the next page...*

## LLWU\_F1 field descriptions (continued)

Field	Description
	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 1 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 wake-up pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 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 1 to WUF3.  0 LLWU_P3 input was not a wake-up source 1 LLWU_P3 input was a wake-up 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 1 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 1 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 wake-up pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF0.  0 LLWU_P0 input was not a wakeup source 1 LLWU_P0 input was a wakeup source

### 17.3.7 LLWU Flag 2 register (LLWU\_F2)

LLWU\_F2 contains the wakeup flags indicating which wakeup source caused the MCU to exit LLS or VLLS mode. For LLS, this is the source causing the CPU interrupt flow. 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 + 6h offset = 4007\_C006h

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\_F2 field descriptions

Field	Description
7 WUF15	<p>Wakeup Flag For LLWU_P15</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF15.</p> <p>0 LLWU_P15 input was not a wakeup source 1 LLWU_P15 input was a wakeup source</p>
6 WUF14	<p>Wakeup Flag For LLWU_P14</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF14.</p> <p>0 LLWU_P14 input was not a wakeup source 1 LLWU_P14 input was a wakeup source</p>
5 WUF13	<p>Wakeup Flag For LLWU_P13</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF13.</p> <p>0 LLWU_P13 input was not a wakeup source 1 LLWU_P13 input was a wakeup source</p>

*Table continues on the next page...*

**LLWU\_F2 field descriptions (continued)**

<b>Field</b>	<b>Description</b>
4 WUF12	<p>Wakeup Flag For LLWU_P12</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF12.</p> <p>0 LLWU_P12 input was not a wakeup source 1 LLWU_P12 input was a wakeup source</p>
3 WUF11	<p>Wakeup Flag For LLWU_P11</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF11.</p> <p>0 LLWU_P11 input was not a wakeup source 1 LLWU_P11 input was a wakeup source</p>
2 WUF10	<p>Wakeup Flag For LLWU_P10</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF10.</p> <p>0 LLWU_P10 input was not a wakeup source 1 LLWU_P10 input was a wakeup source</p>
1 WUF9	<p>Wakeup Flag For LLWU_P9</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF9.</p> <p>0 LLWU_P9 input was not a wakeup source 1 LLWU_P9 input was a wakeup source</p>
0 WUF8	<p>Wakeup Flag For LLWU_P8</p> <p>Indicates that an enabled external wakeup pin was a source of exiting a low-leakage power mode. To clear the flag, write a 1 to WUF8.</p> <p>0 LLWU_P8 input was not a wakeup source 1 LLWU_P8 input was a wakeup source</p>

**17.3.8 LLWU Flag 3 register (LLWU\_F3)**

LLWU\_F3 contains the wakeup flags indicating which internal wakeup source caused the MCU to exit LLS or VLLS mode. For LLS, this is the source causing the CPU interrupt flow. 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 + 7h offset = 4007\_C007h

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\_F3 field descriptions**

Field	Description
7 MWUF7	<p>Wakeup flag For module 7</p> <p>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.</p> <p>0 Module 7 input was not a wakeup source 1 Module 7 input was a wakeup source</p>
6 MWUF6	<p>Wakeup flag For module 6</p> <p>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.</p> <p>0 Module 6 input was not a wakeup source 1 Module 6 input was a wakeup source</p>
5 MWUF5	<p>Wakeup flag For module 5</p> <p>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.</p> <p>0 Module 5 input was not a wakeup source 1 Module 5 input was a wakeup source</p>
4 MWUF4	<p>Wakeup flag For module 4</p> <p>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.</p> <p>0 Module 4 input was not a wakeup source 1 Module 4 input was a wakeup source</p>
3 MWUF3	<p>Wakeup flag For module 3</p> <p>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.</p> <p>0 Module 3 input was not a wakeup source 1 Module 3 input was a wakeup source</p>
2 MWUF2	<p>Wakeup flag For module 2</p>

Table continues on the next page...

**LLWU\_F3 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 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  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

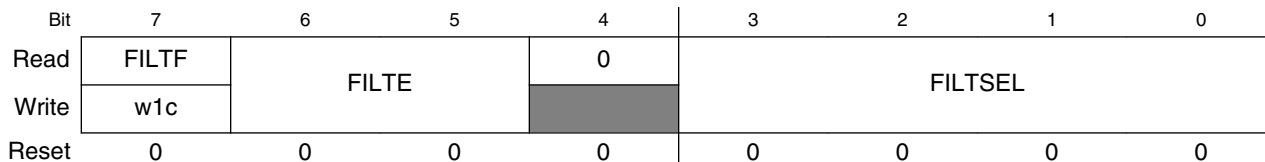
**17.3.9 LLWU Pin Filter 1 register (LLWU\_FILT1)**

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 + 8h offset = 4007\_C008h



**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.

*Table continues on the next page...*

**LLWU\_FILT1 field descriptions (continued)**

Field	Description
	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
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 FILTSEL	Filter Pin Select  Selects 1 out of the 16 wakeup pins to be muxed into the filter.  0000 Select LLWU_P0 for filter ... .. 1111 Select LLWU_P15 for filter

**17.3.10 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 + 9h offset = 4007\_C009h

Bit	7	6	5	4	3	2	1	0
Read	FILTF	FILTE			0	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.

*Table continues on the next page...*

**LLWU\_FILT2 field descriptions (continued)**

Field	Description
	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
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–0 FILTSEL	Filter Pin Select  Selects 1 out of the 16 wakeup pins to be muxed into the filter.  0000 Select LLWU_P0 for filter ... .. 1111 Select LLWU_P15 for filter

## 17.4 Functional description

This 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 LLS and 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. Two 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.

### 17.4.1 LLS mode

Wakeup events triggered from either an external pin input or an internal module interrupt, result in a CPU interrupt flow to begin user code execution.

### 17.4.2 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.

### 17.4.3 Initialization

For an enabled peripheral wakeup input, the peripheral flag must be cleared by software before entering LLS or 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 LLS or VLLSx mode.

After enabling an external pin filter or changing the source pin, wait at least five LPO clock cycles before entering LLS or 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 18

## Miscellaneous Control Module (MCM)

### 18.1 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.

#### 18.1.1 Features

The MCM includes the following features:

- Program-visible information on the platform configuration and revision

### 18.2 Memory map/register descriptions

The memory map and register descriptions below describe the registers using byte addresses.

**MCM memory map**

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
E008_0008	Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)	16	R	000Fh	<a href="#">18.2.1/340</a>
E008_000A	Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)	16	R	0017h	<a href="#">18.2.2/340</a>
E008_000C	Crossbar Switch (AXBS) Control Register (MCM_PLACR)	32	R/W	0000_0000h	<a href="#">18.2.3/341</a>

*Table continues on the next page...*

## MCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
E008_0010	Interrupt Status and Control Register (MCM_ISCR)	32	R	0002_0000h	<a href="#">18.2.4/342</a>
E008_0040	Compute Operation Control Register (MCM_CPO)	32	R/W	0000_0000h	<a href="#">18.2.5/345</a>

## 18.2.1 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: E008\_0000h base + 8h offset = E008\_0008h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Read	0								ASC								
Write	[Shaded]																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	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.
7–0 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 <i>n</i> is absent 1 A bus slave connection to AXBS input port <i>n</i> is present

## 18.2.2 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: E008\_0000h base + Ah offset = E008\_000Ah

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

## MCM\_PLAMC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–0 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 <i>n</i> is absent 1 A bus master connection to AXBS input port <i>n</i> is present

## 18.2.3 Crossbar Switch (AXBS) Control Register (MCM\_PLACR)

The PLACR register selects the arbitration policy for the crossbar masters.

Address: E008\_0000h base + Ch offset = E008\_000Ch

Bit	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R	0																
W	[Shaded]																
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							ARB	Reserved								
W	[Shaded]							[Shaded]	[Shaded]								
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

## MCM\_PLACR field descriptions

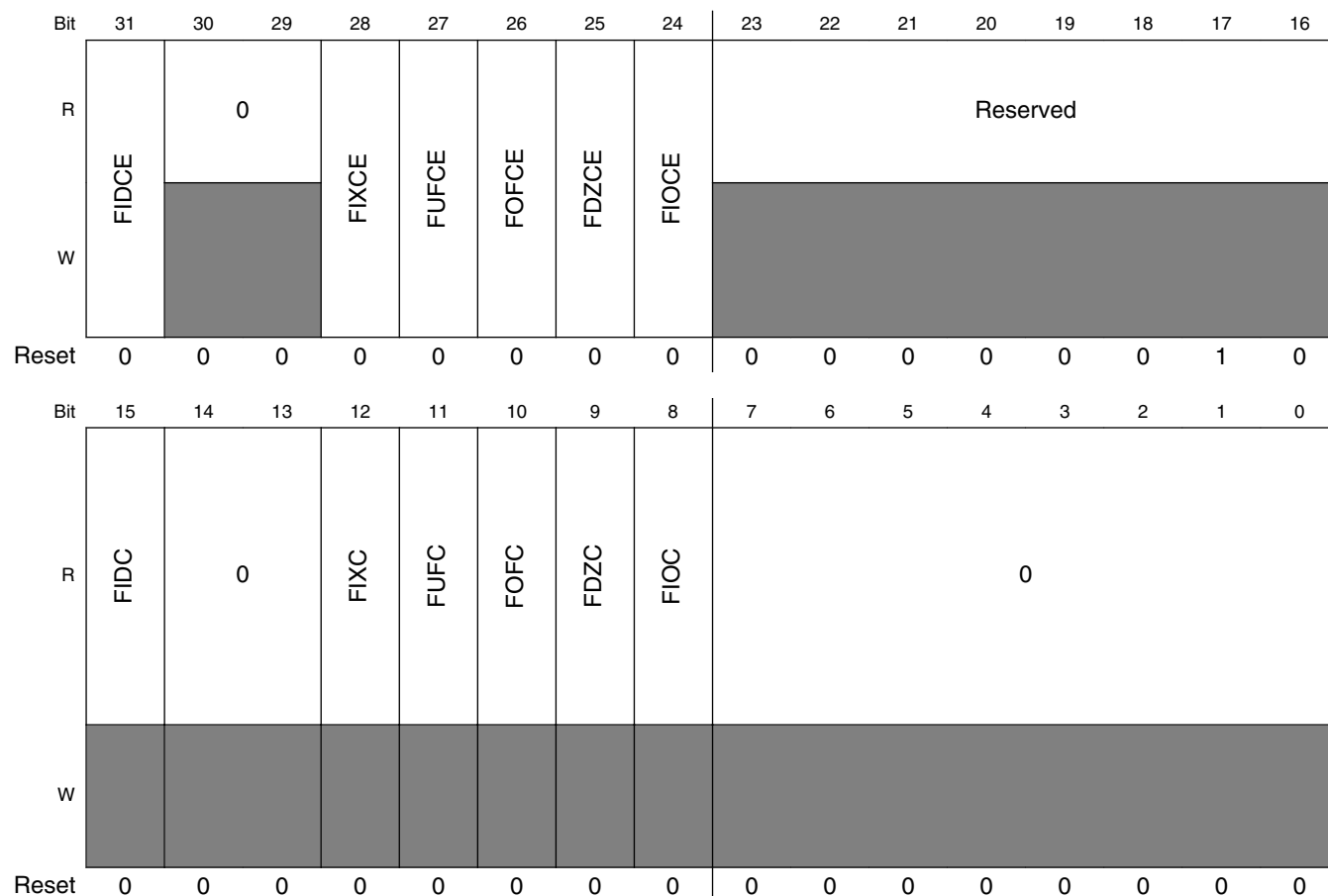
Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 ARB	Arbitration select  0 Fixed-priority arbitration for the crossbar masters 1 Round-robin arbitration for the crossbar masters
8–0 Reserved	This field is reserved.

### 18.2.4 Interrupt Status and Control Register (MCM\_ISCR)

The MCM\_ISCR register includes the enable and status bits associated with the core’s floating-point exceptions. The individual event indicators are first qualified with their exception enables and then logically summed to form an interrupt request sent to the core’s NVIC.

Bits 15-8 are read-only indicator flags based on the processor’s FPSCR register. Attempted writes to these bits are ignored. Once set, the flags remain asserted until software clears the corresponding FPSCR bit.

Address: E008\_0000h base + 10h offset = E008\_0010h



**MCM\_ISCR field descriptions**

Field	Description
31 FIDCE	FPU input denormal interrupt enable 0 Disable interrupt 1 Enable interrupt

Table continues on the next page...

**MCM\_ISCR field descriptions (continued)**

<b>Field</b>	<b>Description</b>
30–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 FIXCE	FPU inexact interrupt enable 0 Disable interrupt 1 Enable interrupt
27 FUFCE	FPU underflow interrupt enable 0 Disable interrupt 1 Enable interrupt
26 FOFCE	FPU overflow interrupt enable 0 Disable interrupt 1 Enable interrupt
25 FDZCE	FPU divide-by-zero interrupt enable 0 Disable interrupt 1 Enable interrupt
24 FIOCE	FPU invalid operation interrupt enable 0 Disable interrupt 1 Enable interrupt
23–16 Reserved	This field is reserved.
15 FIDC	FPU input denormal interrupt status  This read-only bit is a copy of the core's FPSCR[IDC] bit and signals input denormalized number has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IDC] bit.  0 No interrupt 1 Interrupt occurred
14–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 FIXC	FPU inexact interrupt status  This read-only bit is a copy of the core's FPSCR[IXC] bit and signals an inexact number has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IXC] bit.  0 No interrupt 1 Interrupt occurred
11 FUFC	FPU underflow interrupt status  This read-only bit is a copy of the core's FPSCR[UFC] bit and signals an underflow has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[UFC] bit.  0 No interrupt 1 Interrupt occurred
10 FOFC	FPU overflow interrupt status

*Table continues on the next page...*

**MCM\_ISCR field descriptions (continued)**

Field	Description
	<p>This read-only bit is a copy of the core's FPSCR[OFC] bit and signals an overflow has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[OFC] bit.</p> <p>0 No interrupt 1 Interrupt occurred</p>
<p>9 FDZC</p>	<p>FPU divide-by-zero interrupt status</p> <p>This read-only bit is a copy of the core's FPSCR[DZC] bit and signals a divide by zero has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[DZC] bit.</p> <p>0 No interrupt 1 Interrupt occurred</p>
<p>8 FIOC</p>	<p>FPU invalid operation interrupt status</p> <p>This read-only bit is a copy of the core's FPSCR[IOC] bit and signals an illegal operation has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IOC] bit.</p> <p>0 No interrupt 1 Interrupt occurred</p>
<p>7-0 Reserved</p>	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>



## 18.2.5 Compute Operation Control Register (MCM\_CPO)

This register controls the Compute Operation.

Address: E008\_0000h base + 40h offset = E008\_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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	0													CPOWOI	CPOACK	CPOREQ
W	[Reserved]														[Reserved]	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**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 wakeup 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.

**18.3 Functional description**

This section describes the functional description of MCM module.

**18.3.1 Interrupts**

The MCM's interrupt is generated if any of the following is true:

- FPU input denormal interrupt is enabled (FIDCE) and an input is denormalized (FIDC)
- FPU inexact interrupt is enabled (FIXCE) and a number is inexact (FIXC)
- FPU underflow interrupt is enabled (FUFCE) and an underflow occurs (FUFC)
- FPU overflow interrupt is enabled (FOFCE) and an overflow occurs (FOFC)
- FPU divide-by-zero interrupt is enabled (FDZCE) and a divide-by-zero occurs (FDZC)
- FPU invalid operation interrupt is enabled (FDZCE) and an invalid occurs (FDZC)

**18.3.1.1 Determining source of the interrupt**

To determine the exact source of the interrupt qualify the interrupt status flags with the corresponding interrupt enable bits.

1. Form  $MCM\_ISCR[31:16] \&\& MCM\_ISCR[15:0]$
2. Search the result for asserted flags, which indicate the exact interrupt sources

# Chapter 19

## Crossbar Switch Lite (AXBS-Lite)

### 19.1 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.

#### 19.1.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
  - Allows concurrent accesses from different masters to different slaves
- 32-bit data bus
- Operation at a 1-to-1 clock frequency with the bus masters
- Programmable configuration for fixed-priority or round-robin slave port arbitration

## 19.2 Memory Map / Register Definition

This crossbar switch is designed for minimal gate count. It, therefore, has no memory-mapped configuration registers.

## 19.3 Functional Description

### 19.3.1 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.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
  - An outstanding request to one slave port that has a long response time and
  - A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

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 master had to arbitrate to gain control of the slave port.

## 19.3.2 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round-robin

The selection of the global slave port arbitration is controlled by `MCM_PLACR[ARB]`. 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.

### 19.3.2.1 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 19-1. 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: <ul style="list-style-type: none"> <li>• The current master is not running a transfer.</li> <li>• The new requesting master's priority level is higher than that of the current master.</li> </ul>	At the next clock edge
Both of the following are true: <ul style="list-style-type: none"> <li>• The current master is running an undefined length burst transfer.</li> <li>• The requesting master's priority level is higher than that of the current master.</li> </ul>	At the next arbitration point for the undefined length burst transfer <b>NOTE:</b> Arbitration points for an undefined length burst are defined in the MGPCR for each master.
The requesting master's priority level is lower than the current master.	At the conclusion of one of the following cycles: <ul style="list-style-type: none"> <li>• An IDLE cycle</li> <li>• A non-IDLE cycle to a location other than the current slave port</li> </ul>

### 19.3.2.2 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.

## 19.4 Initialization/application information

No initialization is required for the crossbar switch.

See the AXBS section of the configuration chapter for the reset state of the arbitration scheme.





# Chapter 20

## Peripheral Bridge (AIPS-Lite)

### 20.1 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.

#### 20.1.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width

#### 20.1.2 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 peripheral bridge performs a bus protocol conversion of the master transactions and generates the following as inputs to the peripherals:

- Module enables
- Module addresses

## Functional description

- Transfer attributes
- Byte enables
- Write data

The peripheral bridge selects and captures read data from the peripheral interface and returns it to the crossbar switch.

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. Two global external module enables are available for the remaining address space to allow for customization and expansion of addressed peripheral devices.

The AIPS-Lite module uses the data width of accessed peripheral to perform proper data byte lane routing; no bus decomposition (bus sizing) is performed.

## 20.2 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.

### 20.2.1 Access support

Aligned and misaligned 32-bit, 16-bit, and byte accesses are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. All accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted that is larger than the targeted port, an error response is generated.

---

# Chapter 21

## Direct Memory Access Multiplexer (DMAMUX)

### 21.1 Introduction

#### NOTE

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

#### 21.1.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the four DMA channels. This process is illustrated in the following figure.

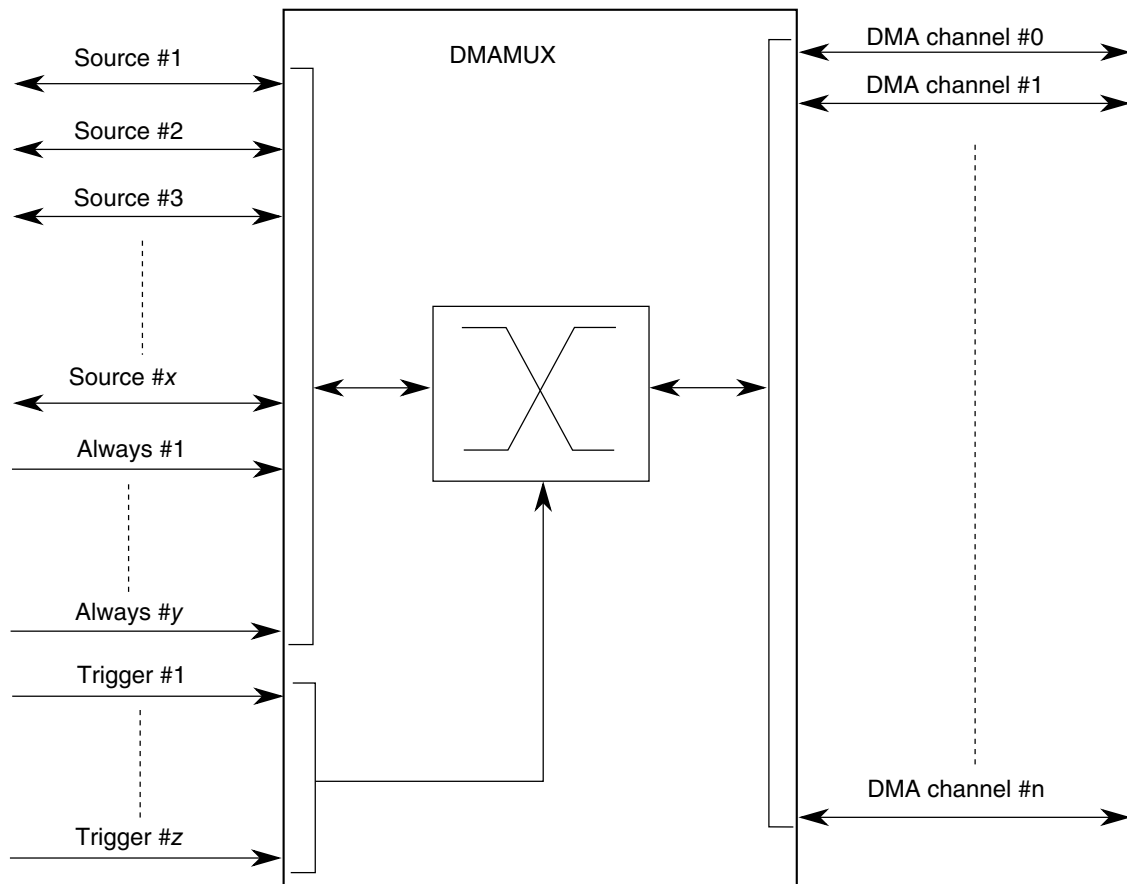


Figure 21-1. DMAMUX block diagram

## 21.1.2 Features

The DMAMUX module provides these features:

- Up to 59 peripheral slots and up to four always-on slots can be routed to four channels.
- four independently selectable DMA channel routers.
  - The first four channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

## 21.1.3 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, for example, changing the period of a DMA trigger.

- 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.

- Periodic Trigger mode

In this mode, a DMA source may only request a DMA transfer, such as when a transmit buffer becomes empty or a receive buffer becomes full, periodically. Configuration of the period is done in the registers of the periodic interrupt timer (PIT). This mode is available only for channels 0–3.

## 21.2 External signal description

The DMAMUX has no external pins.

## 21.3 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	<a href="#">21.3.1/358</a>
4002_1001	Channel Configuration register (DMAMUX_CHCFG1)	8	R/W	00h	<a href="#">21.3.1/358</a>
4002_1002	Channel Configuration register (DMAMUX_CHCFG2)	8	R/W	00h	<a href="#">21.3.1/358</a>
4002_1003	Channel Configuration register (DMAMUX_CHCFG3)	8	R/W	00h	<a href="#">21.3.1/358</a>

### 21.3.1 Channel Configuration register (DMAMUX\_CHCFGn)

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 trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

Address: 4002\_1000h base + 0h offset + (1d × i), where i=0d to 3d

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

#### DMAMUX\_CHCFGn field descriptions

Field	Description
7 ENBL	<p>DMA Channel Enable</p> <p>Enables the DMA channel.</p> <p>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.</p> <p>1 DMA channel is enabled</p>
6 TRIG	<p>DMA Channel Trigger Enable</p> <p>Enables the periodic trigger capability for the triggered DMA channel.</p> <p>0 Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode)</p> <p>1 Triggering is enabled. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode.</p>
5-0 SOURCE	<p>DMA Channel Source (Slot)</p> <p>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.</p>

## 21.4 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.

Functionally, the DMAMUX channels may be divided into two classes:

- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

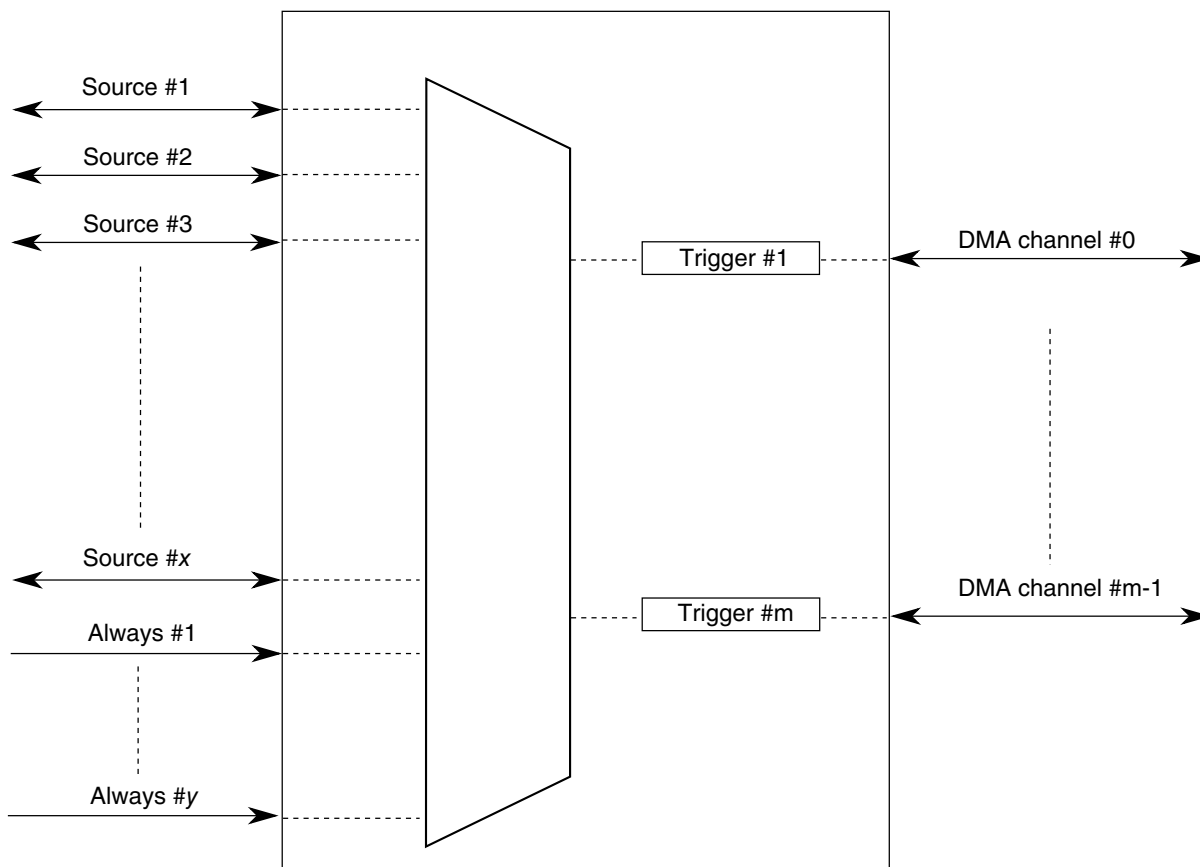
### 21.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 4 channels of the DMAMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention. The trigger is generated by the periodic interrupt timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. See the section on periodic interrupt timer for more information on this topic.

#### Note

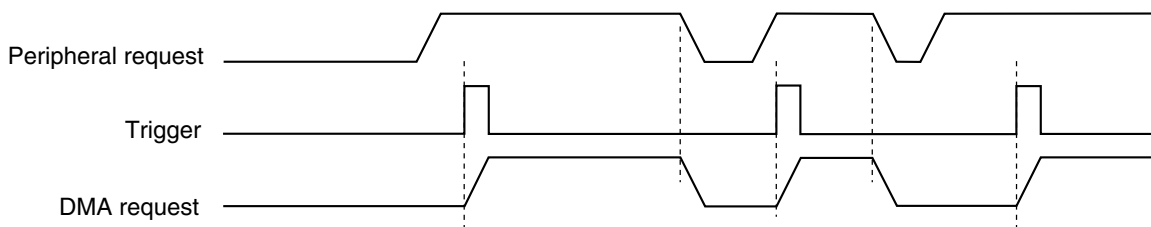
Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

**Functional description**



**Figure 21-7. DMAMUX triggered channels**

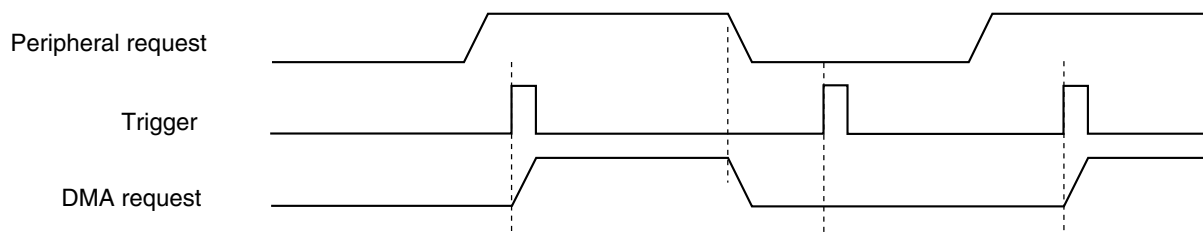
The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in the following figure.



**Figure 21-8. DMAMUX channel triggering: normal operation**

After the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger will be ignored. This situation is illustrated in the following figure.





**Figure 21-9. DMAMUX channel triggering: ignored trigger**

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus

As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. After it has been set up, the SPI will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5  $\mu$ s (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.

- Using the GPIO ports to drive or sample waveforms

By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

## 21.4.2 DMA channels with no triggering capability

The other channels of the DMAMUX provide the normal routing functionality as described in [Modes of operation](#).

### 21.4.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are four 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, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- 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. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

## 21.5 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

### 21.5.1 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.

### 21.5.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields 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. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

#### NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00 to CHCFG1 (base address + 0x01).

2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG1 (base address + 0x01).

The following code example illustrates steps 1 and 4 above:

```
void DMAMUX_Init(uint8_t DMA_CH, uint8_t DMAMUX_SOURCE)
{
    DMAMUX_0.CHCFG[DMA_CH].B.SOURCE = DMAMUX_SOURCE;
    DMAMUX_0.CHCFG[DMA_CH].B.ENBL   = 1;
    DMAMUX_0.CHCFG[DMA_CH].B.TRIG   = 1;
}
```

To enable a source, without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields 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 while CHCFG[TRIG] is cleared.

### NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with no periodic triggering capability:

1. Write 0x00 to CHCFG1 (base address + 0x01).
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1 (base address + 0x01).

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+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
```

```
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
```

```
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] and CHCFG[TRIG] bits 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] and CHCFG[TRIG] fields are 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 (base address + 0x08).
3. Write 0x87 to CHCFG8 (base address + 0x08). (In this example, setting CHCFG[TRIG] would have no effect due to the assumption that channel 8 does not support the periodic triggering functionality.)

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+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
```

## Initialization/application information

```
In File main.c:  
#include "registers.h"  
:  
:  
*CHCFG8 = 0x00;  
*CHCFG8 = 0x87;
```

# Chapter 22

## Enhanced Direct Memory Access (eDMA)

### 22.1 Introduction

#### NOTE

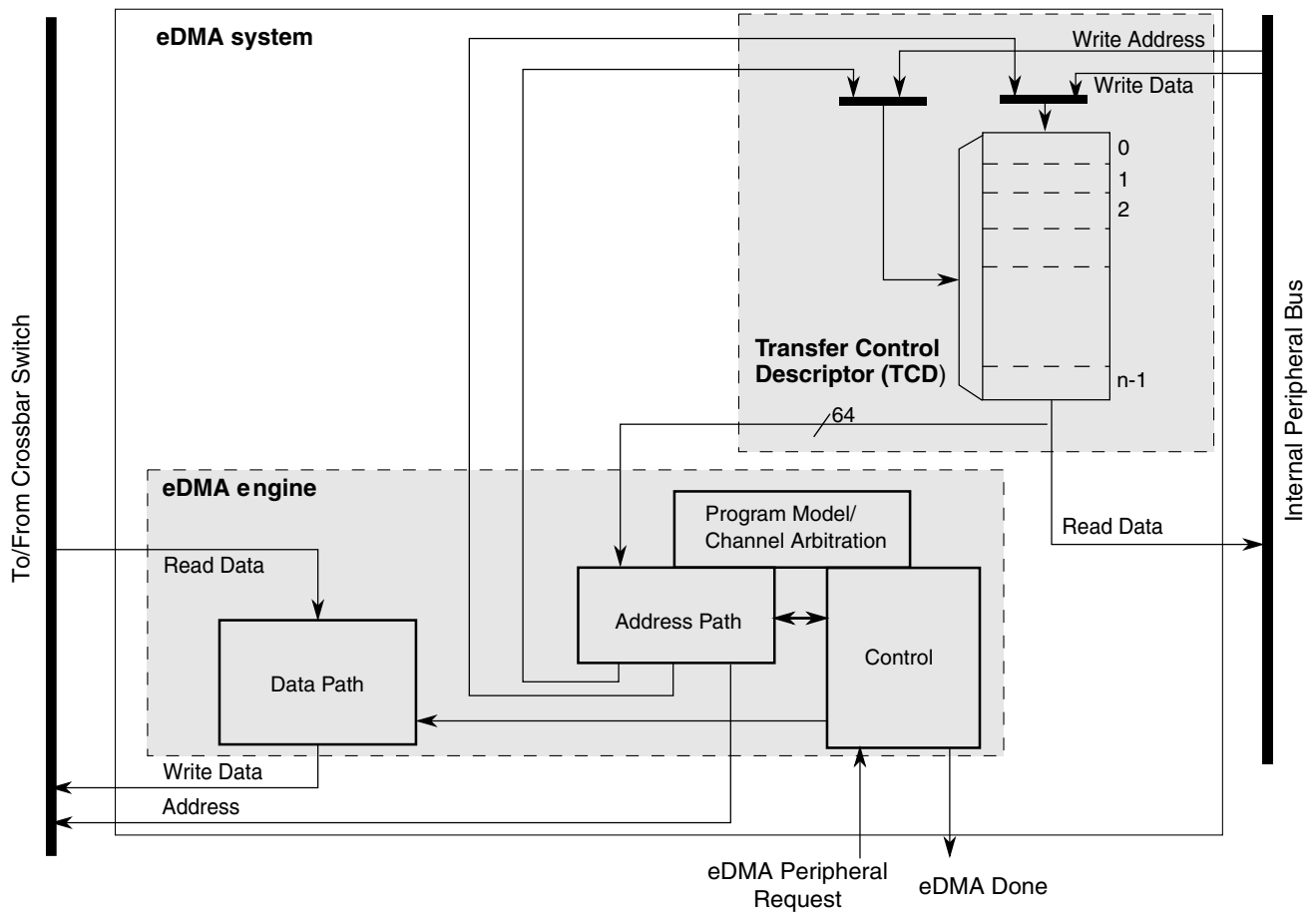
For the chip-specific implementation details of this module's instances, see the chip configuration information.

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- and destination-address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 4 channels

#### 22.1.1 eDMA system block diagram

[Figure 22-1](#) illustrates the components of the eDMA system, including the eDMA module ("engine").



**Figure 22-1. eDMA system block diagram**

### 22.1.2 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 22-1. eDMA engine submodules**

Submodule	Function
Address path	<p>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<sub>n</sub>[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.</p> <p>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 the new values for the TCD<sub>n</sub>{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<sub>n</sub>_CITER field, and a possible fetch of the next TCD<sub>n</sub> from memory as part of a scatter/gather operation.</p>
Data path	<p>This block implements the bus master read/write datapath. It includes 16 bytes of register storage 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.</p> <p>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).</p>
Program model/channel arbitration	<p>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).</p>
Control	<p>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.</p>

The transfer-control descriptor local memory is further partitioned into:

**Table 22-2. Transfer control descriptor memory**

Submodule	Description
Memory controller	<p>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.</p>
Memory array	<p>TCD storage for each channel's transfer profile.</p>

## 22.1.3 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
- 4-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 optional interrupt requests
  - One interrupt per channel, optionally asserted at completion of major iteration count
  - Optional error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Optional support for scatter/gather DMA processing
- Support for complex data structures

- Support to cancel transfers via software error detection and error correction
- Error detection and error correction

In the discussion of this module,  $n$  is used to reference the channel number.

## 22.2 Modes of operation

The eDMA operates in the following modes:

**Table 22-3. 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.  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: <ul style="list-style-type: none"> <li>• If CR[EDBG] is cleared, the DMA continues to operate.</li> <li>• 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.</li> </ul>
Wait	Before entering Wait mode, the DMA attempts to complete its current transfer. After the transfer completes, the device enters Wait mode.

## 22.3 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

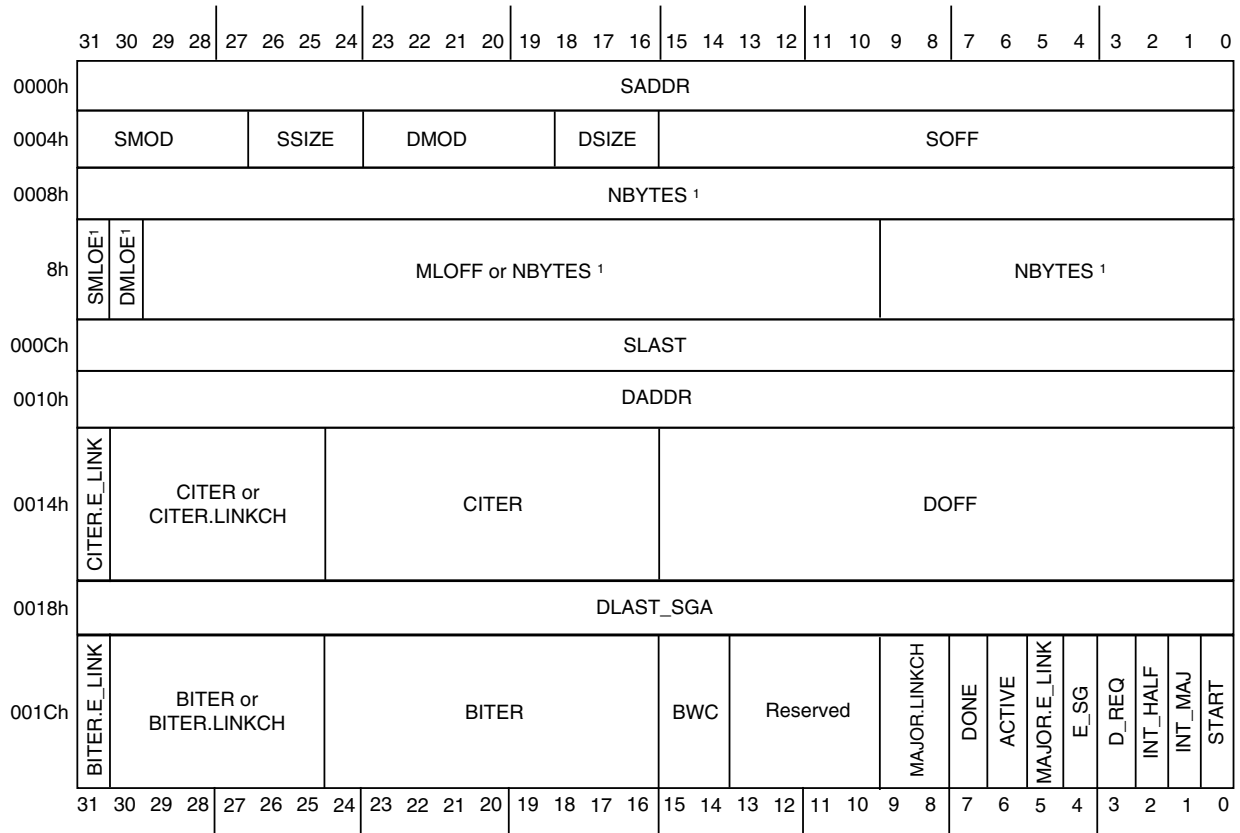
### 22.3.1 TCD memory

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 3. Each TCD $n$  definition is presented as 11 registers of 16 or 32 bits.

## 22.3.2 TCD initialization

Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

### 22.3.3 TCD structure



<sup>1</sup> The fields implemented in Word 2 depend on whether DMA\_CR[EMLM] is 0 or 1.

## 22.3.4 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	22.3.1/377
4000_8004	Error Status Register (DMA_ES)	32	R	0000_0000h	22.3.2/380
4000_800C	Enable Request Register (DMA_ERQ)	32	R/W	0000_0000h	22.3.3/382
4000_8014	Enable Error Interrupt Register (DMA_EEI)	32	R/W	0000_0000h	22.3.4/383
4000_8018	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	22.3.5/384
4000_8019	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	22.3.6/385
4000_801A	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	22.3.7/386
4000_801B	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	22.3.8/387
4000_801C	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	22.3.9/388
4000_801D	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	22.3.10/389
4000_801E	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	22.3.11/390
4000_801F	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	22.3.12/391
4000_8024	Interrupt Request Register (DMA_INT)	32	R/W	0000_0000h	22.3.13/392
4000_802C	Error Register (DMA_ERR)	32	R/W	0000_0000h	22.3.14/393
4000_8034	Hardware Request Status Register (DMA_HRS)	32	R	0000_0000h	22.3.15/394
4000_8044	Enable Asynchronous Request in Stop Register (DMA_EARS)	32	R/W	0000_0000h	22.3.16/396
4000_8100	Channel n Priority Register (DMA_DCHPRI3)	8	R/W	See section	22.3.17/397
4000_8101	Channel n Priority Register (DMA_DCHPRI2)	8	R/W	See section	22.3.17/397
4000_8102	Channel n Priority Register (DMA_DCHPRI1)	8	R/W	See section	22.3.17/397

Table continues on the next page...

## DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4000_8103	Channel n Priority Register (DMA_DCHPRI0)	8	R/W	See section	22.3.17/397
4000_9000	TCD Source Address (DMA_TCD0_SADDR)	32	R/W	Undefined	22.3.18/398
4000_9004	TCD Signed Source Address Offset (DMA_TCD0_SOFF)	16	R/W	Undefined	22.3.19/398
4000_9006	TCD Transfer Attributes (DMA_TCD0_ATTR)	16	R/W	Undefined	22.3.20/399
4000_9008	TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD0_NBYTES_MLNO)	32	R/W	Undefined	22.3.21/400
4000_9008	TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD0_NBYTES_MLOFFNO)	32	R/W	Undefined	22.3.22/400
4000_9008	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD0_NBYTES_MLOFFYES)	32	R/W	Undefined	22.3.23/402
4000_900C	TCD Last Source Address Adjustment (DMA_TCD0_SLAST)	32	R/W	Undefined	22.3.24/403
4000_9010	TCD Destination Address (DMA_TCD0_DADDR)	32	R/W	Undefined	22.3.25/403
4000_9014	TCD Signed Destination Address Offset (DMA_TCD0_DOFF)	16	R/W	Undefined	22.3.26/404
4000_9016	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_CITER_ELINKYES)	16	R/W	Undefined	22.3.27/404
4000_9016	DMA_TCD0_CITER_ELINKNO	16	R/W	Undefined	22.3.28/406
4000_9018	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD0_DLASTSGA)	32	R/W	Undefined	22.3.29/407
4000_901C	TCD Control and Status (DMA_TCD0_CSR)	16	R/W	Undefined	22.3.30/407
4000_901E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_BITER_ELINKYES)	16	R/W	Undefined	22.3.31/410
4000_901E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_BITER_ELINKNO)	16	R/W	Undefined	22.3.32/411
4000_9020	TCD Source Address (DMA_TCD1_SADDR)	32	R/W	Undefined	22.3.18/398
4000_9024	TCD Signed Source Address Offset (DMA_TCD1_SOFF)	16	R/W	Undefined	22.3.19/398
4000_9026	TCD Transfer Attributes (DMA_TCD1_ATTR)	16	R/W	Undefined	22.3.20/399
4000_9028	TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD1_NBYTES_MLNO)	32	R/W	Undefined	22.3.21/400
4000_9028	TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD1_NBYTES_MLOFFNO)	32	R/W	Undefined	22.3.22/400
4000_9028	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD1_NBYTES_MLOFFYES)	32	R/W	Undefined	22.3.23/402
4000_902C	TCD Last Source Address Adjustment (DMA_TCD1_SLAST)	32	R/W	Undefined	22.3.24/403
4000_9030	TCD Destination Address (DMA_TCD1_DADDR)	32	R/W	Undefined	22.3.25/403
4000_9034	TCD Signed Destination Address Offset (DMA_TCD1_DOFF)	16	R/W	Undefined	22.3.26/404
4000_9036	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_CITER_ELINKYES)	16	R/W	Undefined	22.3.27/404
4000_9036	DMA_TCD1_CITER_ELINKNO	16	R/W	Undefined	22.3.28/406

Table continues on the next page...

## DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4000_9038	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD1_DLASTSGA)	32	R/W	Undefined	<a href="#">22.3.29/407</a>
4000_903C	TCD Control and Status (DMA_TCD1_CSR)	16	R/W	Undefined	<a href="#">22.3.30/407</a>
4000_903E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_BITER_ELINKYES)	16	R/W	Undefined	<a href="#">22.3.31/410</a>
4000_903E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_BITER_ELINKNO)	16	R/W	Undefined	<a href="#">22.3.32/411</a>
4000_9040	TCD Source Address (DMA_TCD2_SADDR)	32	R/W	Undefined	<a href="#">22.3.18/398</a>
4000_9044	TCD Signed Source Address Offset (DMA_TCD2_SOFF)	16	R/W	Undefined	<a href="#">22.3.19/398</a>
4000_9046	TCD Transfer Attributes (DMA_TCD2_ATTR)	16	R/W	Undefined	<a href="#">22.3.20/399</a>
4000_9048	TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD2_NBYTES_MLNO)	32	R/W	Undefined	<a href="#">22.3.21/400</a>
4000_9048	TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD2_NBYTES_MLOFFNO)	32	R/W	Undefined	<a href="#">22.3.22/400</a>
4000_9048	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD2_NBYTES_MLOFFYES)	32	R/W	Undefined	<a href="#">22.3.23/402</a>
4000_904C	TCD Last Source Address Adjustment (DMA_TCD2_SLAST)	32	R/W	Undefined	<a href="#">22.3.24/403</a>
4000_9050	TCD Destination Address (DMA_TCD2_DADDR)	32	R/W	Undefined	<a href="#">22.3.25/403</a>
4000_9054	TCD Signed Destination Address Offset (DMA_TCD2_DOFF)	16	R/W	Undefined	<a href="#">22.3.26/404</a>
4000_9056	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_CITER_ELINKYES)	16	R/W	Undefined	<a href="#">22.3.27/404</a>
4000_9056	DMA_TCD2_CITER_ELINKNO	16	R/W	Undefined	<a href="#">22.3.28/406</a>
4000_9058	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD2_DLASTSGA)	32	R/W	Undefined	<a href="#">22.3.29/407</a>
4000_905C	TCD Control and Status (DMA_TCD2_CSR)	16	R/W	Undefined	<a href="#">22.3.30/407</a>
4000_905E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_BITER_ELINKYES)	16	R/W	Undefined	<a href="#">22.3.31/410</a>
4000_905E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_BITER_ELINKNO)	16	R/W	Undefined	<a href="#">22.3.32/411</a>
4000_9060	TCD Source Address (DMA_TCD3_SADDR)	32	R/W	Undefined	<a href="#">22.3.18/398</a>
4000_9064	TCD Signed Source Address Offset (DMA_TCD3_SOFF)	16	R/W	Undefined	<a href="#">22.3.19/398</a>
4000_9066	TCD Transfer Attributes (DMA_TCD3_ATTR)	16	R/W	Undefined	<a href="#">22.3.20/399</a>
4000_9068	TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD3_NBYTES_MLNO)	32	R/W	Undefined	<a href="#">22.3.21/400</a>
4000_9068	TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD3_NBYTES_MLOFFNO)	32	R/W	Undefined	<a href="#">22.3.22/400</a>
4000_9068	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD3_NBYTES_MLOFFYES)	32	R/W	Undefined	<a href="#">22.3.23/402</a>

Table continues on the next page...



## DMA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4000_906C	TCD Last Source Address Adjustment (DMA_TCD3_SLAST)	32	R/W	Undefined	<a href="#">22.3.24/403</a>
4000_9070	TCD Destination Address (DMA_TCD3_DADDR)	32	R/W	Undefined	<a href="#">22.3.25/403</a>
4000_9074	TCD Signed Destination Address Offset (DMA_TCD3_DOFF)	16	R/W	Undefined	<a href="#">22.3.26/404</a>
4000_9076	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_CITER_ELINKYES)	16	R/W	Undefined	<a href="#">22.3.27/404</a>
4000_9076	DMA_TCD3_CITER_ELINKNO	16	R/W	Undefined	<a href="#">22.3.28/406</a>
4000_9078	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD3_DLASTSGA)	32	R/W	Undefined	<a href="#">22.3.29/407</a>
4000_907C	TCD Control and Status (DMA_TCD3_CSR)	16	R/W	Undefined	<a href="#">22.3.30/407</a>
4000_907E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_BITER_ELINKYES)	16	R/W	Undefined	<a href="#">22.3.31/410</a>
4000_907E	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_BITER_ELINKNO)	16	R/W	Undefined	<a href="#">22.3.32/411</a>

### 22.3.1 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.

**Memory map/register definition**

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ACTIVE	0				Reserved				0				CX	ECX	
W	[Shaded]															
Reset	0	0	0	0	x*	x*	x*	x*	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								EMLM	CLM	HALT	HOE	Reserved	ERCA	EDBG	Reserved
W	[Shaded]								EMLM	CLM	HALT	HOE	Reserved	ERCA	EDBG	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- \* 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.
30–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–24 Reserved	This field is 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 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.

Table continues on the next page...

**DMA\_CR field descriptions (continued)**

Field	Description
3 Reserved	This field is 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.

**22.3.2 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

See the Error Reporting and Handling section for more details.

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	[Shaded]															
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	[Shaded]															
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

*Table continues on the next page...*

## DMA\_ES field descriptions (continued)

Field	Description
	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.
13–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 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 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. <ul style="list-style-type: none"> <li>• TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or</li> <li>• TCDn_CITER[CITER] is equal to zero, or</li> <li>• TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]</li> </ul>

Table continues on the next page...

**DMA\_ES field descriptions (continued)**

Field	Description
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

**22.3.3 Enable Request Register (DMA\_ERQ)**

The ERQ register provides a bit map for the 4 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	[Shaded]															
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												ERQ3	ERQ2	ERQ1	ERQ0
W	[Shaded]												[Shaded]	[Shaded]	[Shaded]	[Shaded]
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**DMA\_ERQ field descriptions**

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

*Table continues on the next page...*

**DMA\_ERQ field descriptions (continued)**

Field	Description
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

**22.3.4 Enable Error Interrupt Register (DMA\_EEI)**

The EEI register provides a bit map for the 4 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												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
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

*Table continues on the next page...*

## DMA\_EEI field descriptions (continued)

Field	Description
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
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

## 22.3.5 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

Table continues on the next page...



**DMA\_CEEI field descriptions (continued)**

Field	Description
	0 Clear only the EEI bit specified in the CEEI field 1 Clear all bits in EEI
5–2 Reserved	This field is reserved.
1–0 CEEI	Clear Enable Error Interrupt  Clears the corresponding bit in EEI

**22.3.6 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
5–2 Reserved	This field is reserved.
1–0 SEEI	Set Enable Error Interrupt  Sets the corresponding bit in EEI

### 22.3.7 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				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
5-2 Reserved	This field is reserved.
1-0 CERQ	Clear Enable Request Clears the corresponding bit in ERQ.

## 22.3.8 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
5-2 Reserved	This field is reserved.
1-0 SERQ	Set Enable Request Sets the corresponding bit in ERQ.

### 22.3.9 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]
5-2 Reserved	This field is reserved.
1-0 CDNE	Clear DONE Bit Clears the corresponding bit in TCDn_CSR[DONE]

### 22.3.10 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]
5-2 Reserved	This field is reserved.
1-0 SSRT	Set START Bit Sets the corresponding bit in TCDn_CSR[START]

### 22.3.11 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
5-2 Reserved	This field is reserved.
1-0 CERR	Clear Error Indicator Clears the corresponding bit in ERR

### 22.3.12 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
5-2 Reserved	This field is reserved.
1-0 CINT	Clear Interrupt Request Clears the corresponding bit in INT

### 22.3.13 Interrupt Request Register (DMA\_INT)

The INT register provides a bit map for the 4 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 (INTC). 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	[Shaded]															
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												INT3	INT2	INT1	INT0
W	[Shaded]												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
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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

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
0 INT0	Interrupt Request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

## 22.3.14 Error Register (DMA\_ERR)

The ERR provides a bit map for the 4 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.

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												ERR3	ERR2	ERR1	ERR0
W													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
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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

### 22.3.15 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

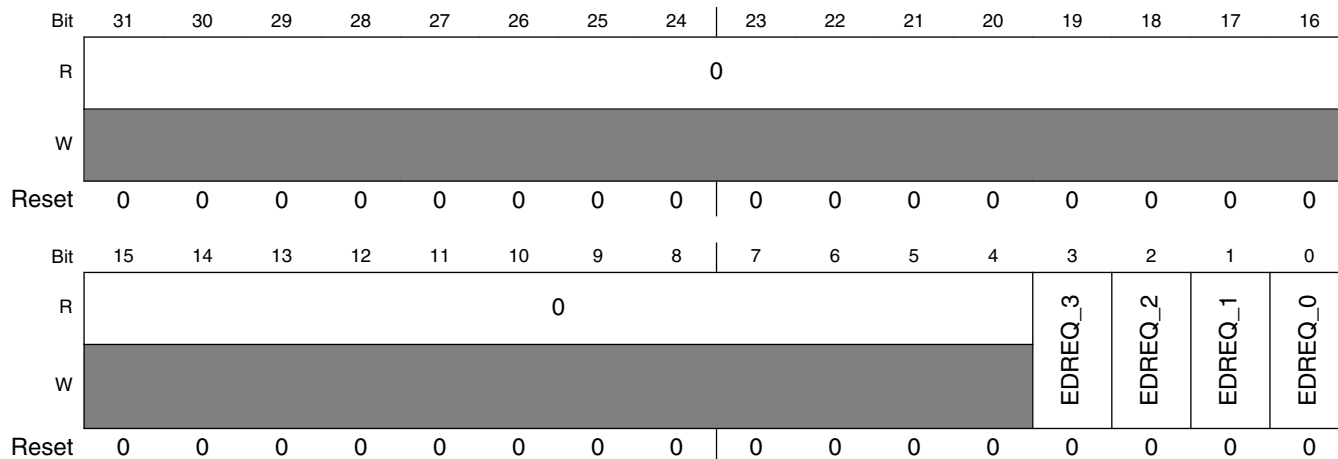
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	[Shaded]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												HRS3	HRS2	HRS1	HRS0
W	[Shaded]												[Shaded]	[Shaded]	[Shaded]	[Shaded]
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**DMA\_HRS field descriptions**

<b>Field</b>	<b>Description</b>
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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

### 22.3.16 Enable Asynchronous Request in Stop Register (DMA\_EARS)

Address: 4000\_8000h base + 44h offset = 4000\_8044h



#### DMA\_EARS field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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. 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.

### 22.3.17 Channel n Priority Register (DMA\_DCHPRIn)

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 3.

Address:  $4000\_8000h$  base +  $100h$  offset +  $(1d \times i)$ , where  $i=0d$  to  $3d$

Bit	7	6	5	4	3	2	1	0
Read	ECP	DPA	0				CHPRI	
Write								
Reset	0	0	0	0	0	0	*	*

\* Notes:

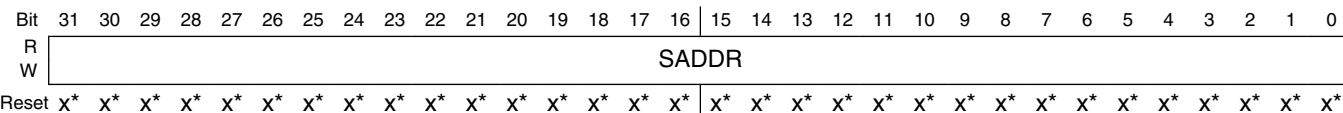
- CHPRI field: See bit field description.

#### DMA\_DCHPRIn 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.
5–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–0 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled <b>NOTE:</b> Reset value for the channel priority field, CHPRI, is equal to the corresponding channel number for each priority register, that is, $DCHPRI3[CHPRI] = 0b11$ .

### 22.3.18 TCD Source Address (DMA\_TCDn\_SADDR)

Address: 4000\_8000h base + 1000h offset + (32d × i), where i=0d to 3d



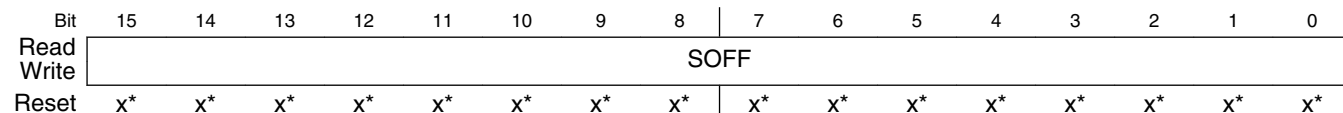
- \* Notes:
- x = Undefined at reset.

#### DMA\_TCDn\_SADDR field descriptions

Field	Description
31–0 SADDR	Source Address Memory address pointing to the source data.

### 22.3.19 TCD Signed Source Address Offset (DMA\_TCDn\_SOFF)

Address: 4000\_8000h base + 1004h offset + (32d × i), where i=0d to 3d



- \* Notes:
- x = Undefined at reset.

#### DMA\_TCDn\_SOFF field descriptions

Field	Description
15–0 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.

## 22.3.20 TCD Transfer Attributes (DMA\_TCDn\_ATTR)

Address: 4000\_8000h base + 1006h offset + (32d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Read	SMOD								SSIZE				DMOD				DSIZE			
Write																				
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*				

\* Notes:

- x = Undefined at reset.

### DMA\_TCDn\_ATTR field descriptions

Field	Description
15–11 SMOD	<p>Source Address Modulo</p> <p>0 Source address modulo feature is disabled</p> <p>≠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.</p>
10–8 SSIZE	<p>Source data transfer size</p> <p><b>NOTE:</b> Using a Reserved value causes a configuration error.</p> <p>000 8-bit</p> <p>001 16-bit</p> <p>010 32-bit</p> <p>011 Reserved</p> <p>100 16-byte</p> <p>101 32-byte burst (4 beats of 64 bits)</p> <p>110 Reserved</p> <p>111 Reserved</p>
7–3 DMOD	<p>Destination Address Modulo</p> <p>See the SMOD definition</p>
2–0 DSIZE	<p>Destination data transfer size</p> <p>See the SSIZE definition</p>

### 22.3.21 TCD Minor Byte Count (Minor Loop Disabled) (DMA\_TCDn\_NBYTES\_MLNO)

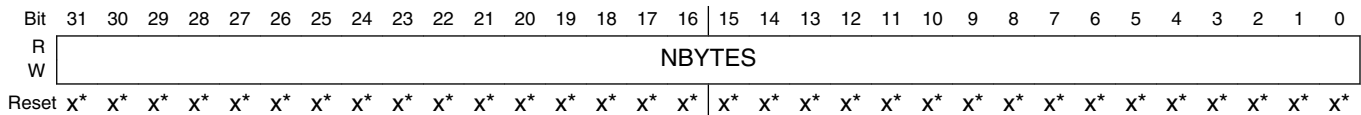
This register, or one of the next two registers (TCD\_NBYTES\_MLOFFNO, TCD\_NBYTES\_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

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 3d



\* Notes:

- x = Undefined at reset.

#### DMA\_TCDn\_NBYTES\_MLNO field descriptions

Field	Description
31–0 NBYTES	<p>Minor Byte Transfer Count</p> <p>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.</p> <p><b>NOTE:</b> An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.</p>

### 22.3.22 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA\_TCDn\_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 3d

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

\* Notes:

- x = Undefined at reset.

### DMA\_TCDn\_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
29–0 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.

### 22.3.23 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA\_TCDn\_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 3d

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

\* Notes:

- x = Undefined at reset.

#### DMA\_TCDn\_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\_TCDn\_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.
9–0 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.

## 22.3.24 TCD Last Source Address Adjustment (DMA\_TCDn\_SLAST)

Address: 4000\_8000h base + 100Ch offset + (32d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W																																
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\_TCDn\_SLAST field descriptions

Field	Description
31–0 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.

## 22.3.25 TCD Destination Address (DMA\_TCDn\_DADDR)

Address: 4000\_8000h base + 1010h offset + (32d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																																
W																																
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\_TCDn\_DADDR field descriptions

Field	Description
31–0 DADDR	Destination Address Memory address pointing to the destination data.

### 22.3.26 TCD Signed Destination Address Offset (DMA\_TCDn\_DOFF)

Address: 4000\_8000h base + 1014h offset + (32d × i), where i=0d to 3d

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

\* Notes:

- x = Undefined at reset.

### DMA\_TCDn\_DOFF field descriptions

Field	Description
15–0 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.

### 22.3.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA\_TCDn\_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 3d

Bit	15	14	13	12	11	10	9	8	
Read	ELINK		0				LINKCH		CITER
Write									
Reset	x*	x*	x*	x*	x*	x*	x*	x*	

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

\* Notes:

- x = Undefined at reset.

## DMA\_TCDn\_CITER\_ELINKYES field descriptions

Field	Description
15 ELINK	<p>Enable channel-to-channel linking on minor-loop complete</p> <p>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.</p> <p>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.</p> <p><b>NOTE:</b> This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported.</p> <p>0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled</p>
14–11 Reserved	This field is reserved.
10–9 LINKCH	<p>Link Channel Number</p> <p>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.</p>
8–0 CITER	<p>Current Major Iteration Count</p> <p>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.</p> <p><b>NOTE:</b> When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field.</p> <p><b>NOTE:</b> If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.</p>

### 22.3.28 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA\_TCDn\_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 3d

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

\* Notes:

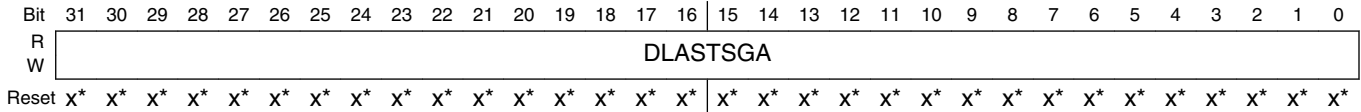
- x = Undefined at reset.

#### DMA\_TCDn\_CITER\_ELINKNO field descriptions

Field	Description
15 ELINK	<p>Enable channel-to-channel linking on minor-loop complete</p> <p>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.</p> <p>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.</p> <p><b>NOTE:</b> This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported.</p> <p>0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled</p>
14–0 CITER	<p>Current Major Iteration Count</p> <p>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.</p> <p><b>NOTE:</b> When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field.</p> <p><b>NOTE:</b> If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.</p>

### 22.3.29 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA\_TCDn\_DLASTSGA)

Address: 4000\_8000h base + 1018h offset + (32d × i), where i=0d to 3d



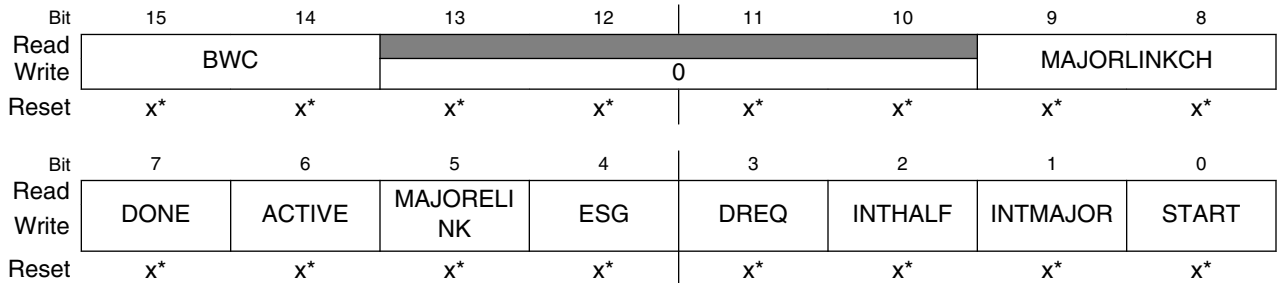
- \* Notes:
- x = Undefined at reset.

#### DMA\_TCDn\_DLASTSGA field descriptions

Field	Description
31–0 DLASTSGA	<p>Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather).</p> <p>If (TCDn_CSR[ESG] = 0) then:</p> <ul style="list-style-type: none"> <li>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.</li> <li>This field uses two's complement notation for the final destination address adjustment.</li> </ul> <p>Otherwise:</p> <ul style="list-style-type: none"> <li>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.</li> </ul>

### 22.3.30 TCD Control and Status (DMA\_TCDn\_CSR)

Address: 4000\_8000h base + 101Ch offset + (32d × i), where i=0d to 3d



- \* Notes:
- x = Undefined at reset.

## DMA\_TCDn\_CSR field descriptions

Field	Description
15–14 BWC	<p>Bandwidth Control</p> <p>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.</p> <p><b>NOTE:</b> If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency.</p> <p>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.</p>
13–10 Reserved	This field is reserved.
9–8 MAJORLINKCH	<p>Link Channel Number</p> <p>If (MAJORELINK = 0) then:</p> <ul style="list-style-type: none"> <li>No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted.</li> </ul> <p>Otherwise:</p> <ul style="list-style-type: none"> <li>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.</li> </ul>
7 DONE	<p>Channel Done</p> <p>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.</p> <p><b>NOTE:</b> This bit must be cleared to write the MAJORELINK or ESG bits.</p>
6 ACTIVE	<p>Channel Active</p> <p>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.</p>
5 MAJORELINK	<p>Enable channel-to-channel linking on major loop complete</p> <p>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.</p> <p><b>NOTE:</b> To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set.</p> <p>0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.</p>
4 ESG	<p>Enable Scatter/Gather Processing</p> <p>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.</p> <p><b>NOTE:</b> 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.</p>

*Table continues on the next page...*



## DMA\_TCDn\_CSR field descriptions (continued)

Field	Description
	<p>0 The current channel's TCD is normal format.</p> <p>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.</p>
3 DREQ	<p>Disable Request</p> <p>If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero.</p> <p>0 The channel's ERQ bit is not affected.</p> <p>1 The channel's ERQ bit is cleared when the major loop is complete.</p>
2 INTHALF	<p>Enable an interrupt when major counter is half complete.</p> <p>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 <math>(CITER == (BITER \gg 1))</math>. 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.</p> <p><b>NOTE:</b> If <math>BITER = 1</math>, do not use INTHALF. Use INTMAJOR instead.</p> <p>0 The half-point interrupt is disabled.</p> <p>1 The half-point interrupt is enabled.</p>
1 INTMAJOR	<p>Enable an interrupt when major iteration count completes.</p> <p>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.</p> <p>0 The end-of-major loop interrupt is disabled.</p> <p>1 The end-of-major loop interrupt is enabled.</p>
0 START	<p>Channel Start</p> <p>If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution.</p> <p>0 The channel is not explicitly started.</p> <p>1 The channel is explicitly started via a software initiated service request.</p>

## 22.3.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA\_TCDn\_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 3d

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

\* Notes:

- x = Undefined at reset.

### DMA\_TCDn\_BITER\_ELINKYES field descriptions

Field	Description
15 ELINK	<p>Enables channel-to-channel linking on minor loop complete</p> <p>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.</p> <p><b>NOTE:</b> 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.</p> <p>0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled</p>
14–11 Reserved	This field is reserved.
10–9 LINKCH	<p>Link Channel Number</p> <p>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.</p> <p><b>NOTE:</b> 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.</p>
8–0 BITER	<p>Starting major iteration count</p> <p>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.</p>

Table continues on the next page...

## DMA\_TCDn\_BITER\_ELINKYES field descriptions (continued)

Field	Description
	<b>NOTE:</b> 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.

### 22.3.32 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA\_TCDn\_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 3d

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

\* Notes:

- x = Undefined at reset.

### DMA\_TCDn\_BITER\_ELINKNO field descriptions

Field	Description
15 ELINK	<p>Enables channel-to-channel linking on minor loop complete</p> <p>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.</p> <p><b>NOTE:</b> 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.</p> <p>0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled</p>
14–0 BITER	<p>Starting Major Iteration Count</p> <p>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.</p> <p><b>NOTE:</b> 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</p>

Table continues on the next page...

**DMA\_TCDn\_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.

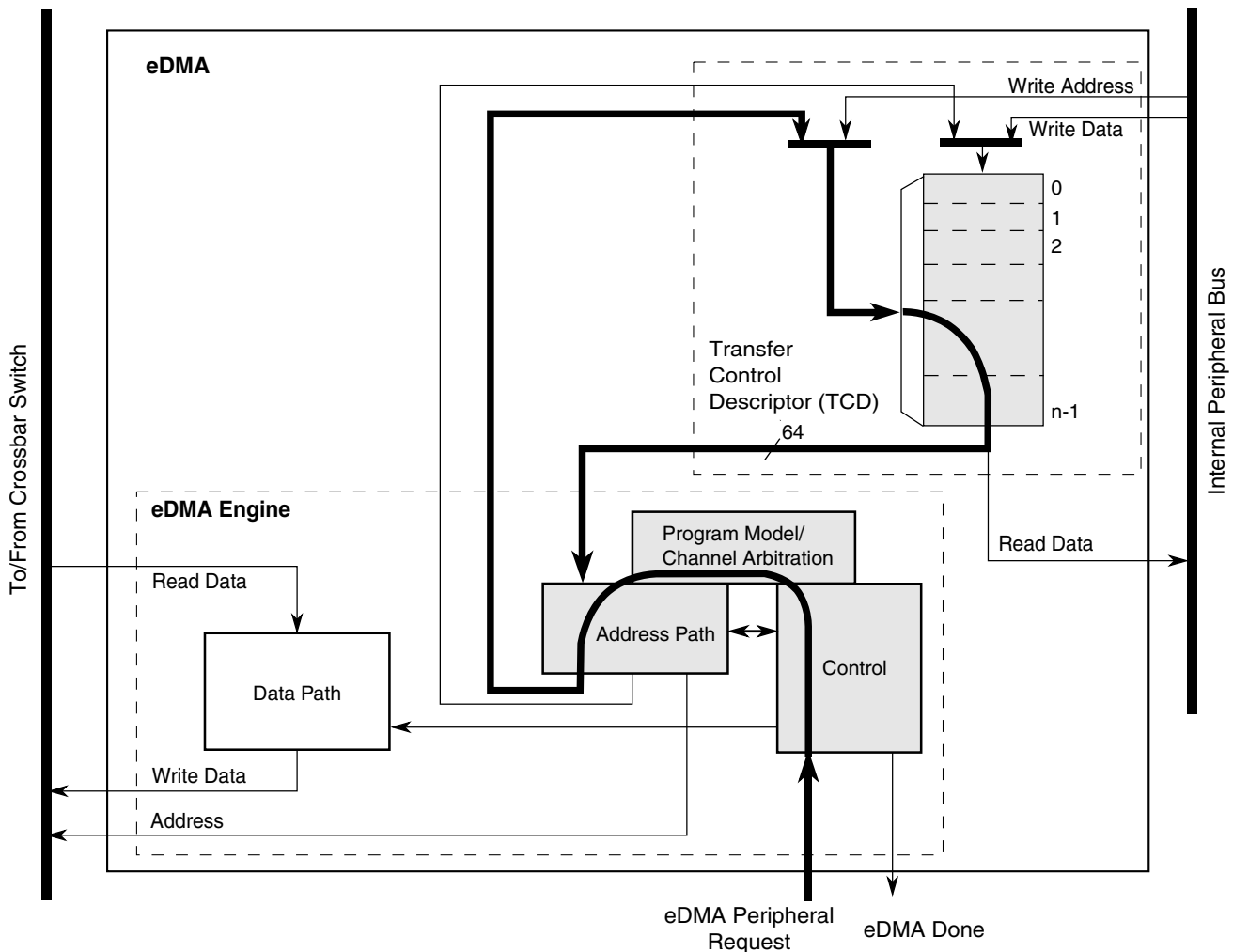
## 22.4 Functional description

The operation of the eDMA is described in the following subsections.

### 22.4.1 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 22-98. 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  $TCDn\_CSR[START]$  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  $TCDn$ . 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:

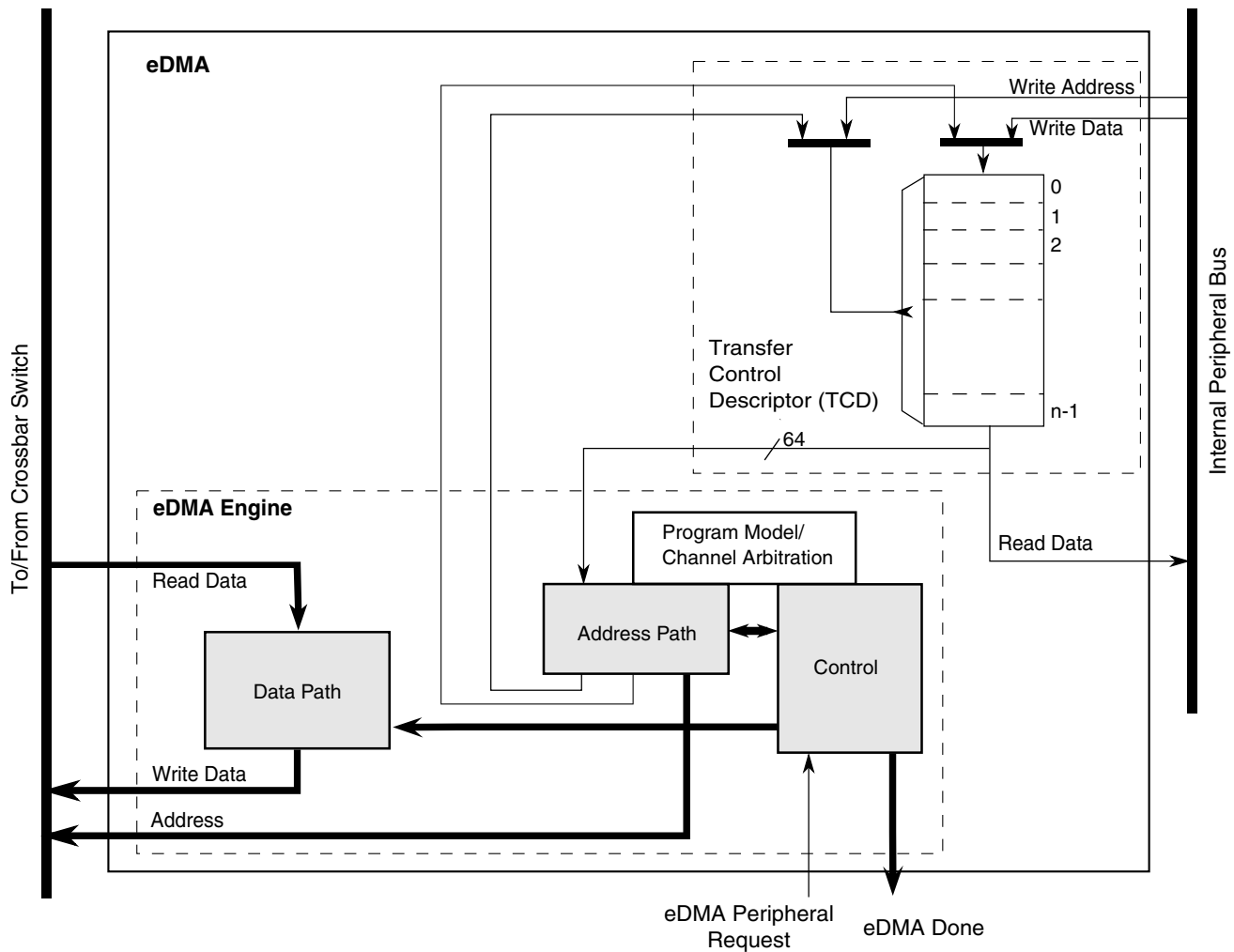


Figure 22-99. eDMA operation, part 2

## Functional description

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, e.g., 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.

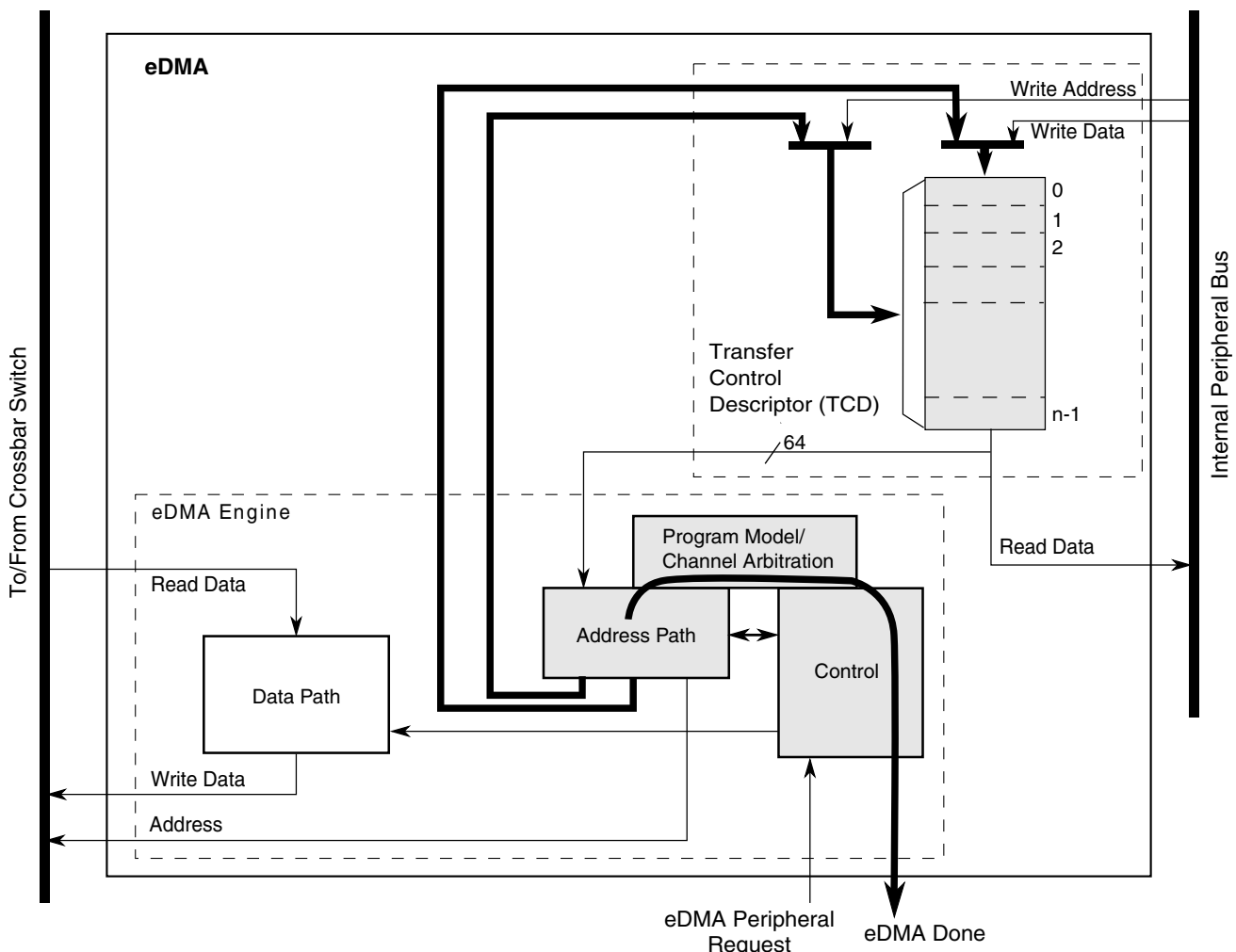


Figure 22-100. eDMA operation, part 3

## 22.4.2 Fault reporting and handling

Channel errors are reported in the Error Status register (DMA<sub>x</sub>\_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.
- 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 TCD<sub>n</sub>\_CITER[E\_LINK] bit does not equal the TCD<sub>n</sub>\_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.

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.

### **22.4.3 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.

## 22.4.4 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.

### 22.4.4.1 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

This table compares peak transfer rates based on different possible system speeds. Specific chips/devices may not support all system speeds listed.

**Table 22-101. 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
66.7 MHz, 32 bit	133.3	66.7	53.3
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.

#### 22.4.4.2 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 22-102. 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 <sub>n</sub> _CSR[START] bit initiated requests start at this point with the registering of the user write to TCD <sub>n</sub> word 7.
3		Channel arbitration begins.
4		Channel arbitration completes. The transfer control descriptor local memory read is initiated.

*Table continues on the next page...*

Table 22-102. Hardware service request process (continued)

Cycle		Description
With internal peripheral bus read and internal SRAM write	With SRAM read and internal peripheral bus write	
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.
13	14	The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD <sub>n</sub> fields into the local memory. The TCD <sub>n</sub> 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 <sub>n</sub> are written back into the local memory.
15	16	The fields in the second part of the TCD <sub>n</sub> 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 22-103. eDMA peak request rate (MReq/sec)

System frequency (MHz)	Request rate with zero wait states	Request rate with wait states
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

## Functional description

A general formula to compute the peak request rate with overlapping requests is:

$$\text{PEAKreq} = \text{freq} / [ \text{entry} + (1 + \text{read\_ws}) + (1 + \text{write\_ws}) + \text{exit} ]$$

where:

**Table 22-104. 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)

### 22.4.4.3 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,

$$\text{PEAKreq} = 150 \text{ MHz} / [ 4 + (1 + 1) + (1 + 3) + 3 ] \text{ cycles} = 11.5 \text{ Mreq/sec}$$

For an internal peripheral bus to SRAM transfer,

$$\text{PEAKreq} = 150 \text{ MHz} / [ 4 + (1 + 2) + (1 + 1) + 3 ] \text{ cycles} = 12.5 \text{ Mreq/sec}$$

Assuming an even distribution of the two transfer types, the average peak request rate would be:

$$\text{PEAKreq} = (11.5 \text{ Mreq/sec} + 12.5 \text{ Mreq/sec}) / 2 = 12.0 \text{ Mreq/sec}$$

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.

## 22.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

### 22.5.1 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 ERQH and ERQL registers.
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 `TCDn_SADDR`, to the destination, as defined by `TCDn_DADDR`, continue until the number of bytes specified by `TCDn_NBYTES` are transferred.

When the transfer is complete, the eDMA engine's local `TCDn_SADDR`, `TCDn_DADDR`, and `TCDn_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 22-105. TCD Control and Status fields**

TCDn_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).

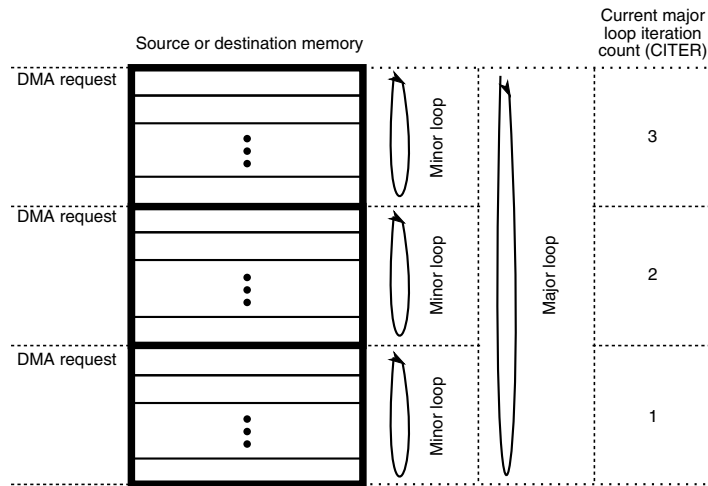


Figure 22-101. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

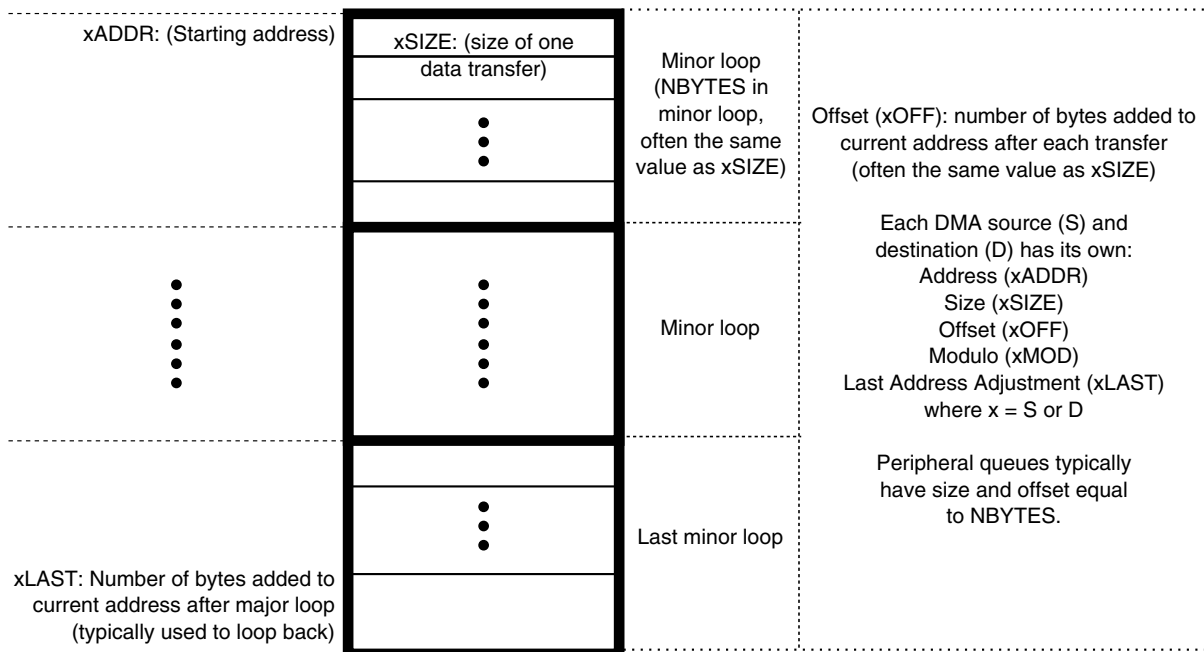


Figure 22-102. Memory array terms

### 22.5.2 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 (DMA<sub>x</sub>\_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.

### **22.5.3 Arbitration mode considerations**

This section discusses arbitration considerations for the eDMA.

#### **22.5.3.1 Fixed channel arbitration**

In this mode, the channel service request from the highest priority channel is selected to execute.

#### **22.5.3.2 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.

### **22.5.4 Performing DMA transfers**

This section presents examples on how to perform DMA transfers with the eDMA.

#### **22.5.4.1 Single request**

To perform a simple transfer of  $n$  bytes of data with one activation, set the major loop to one ( $TCDn\_CITER = TCDn\_BITER = 1$ ). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the  $TCDn\_CSR[DONE]$  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 TCDn\_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCDn\_CSR[DONE] = 0, TCDn\_CSR[START] = 0, TCDn\_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:  $TCDn\_SADDR = 0x1000$ ,  $TCDn\_DADDR = 0x2000$ ,  $TCDn\_CITER = 1$  ( $TCDn\_BITER$ ).
7. The eDMA engine writes:  $TCDn\_CSR[ACTIVE] = 0$ ,  $TCDn\_CSR[DONE] = 1$ ,  $INT[n] = 1$ .
8. The channel retires and the eDMA goes idle or services the next channel.

### 22.5.4.2 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:  $TCDn\_CSR[DONE] = 0$ ,  $TCDn\_CSR[START] = 0$ ,  $TCDn\_CSR[ACTIVE] = 1$ .
4. eDMA engine reads: channel  $TCDn$  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:  $TCDn\_SADDR = 0x1010$ ,  $TCDn\_DADDR = 0x2010$ ,  $TCDn\_CITER = 1$ .
  7. eDMA engine writes:  $TCDn\_CSR[ACTIVE] = 0$ .
  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:  $TCDn\_CSR[DONE] = 0$ ,  $TCDn\_CSR[START] = 0$ ,  $TCDn\_CSR[ACTIVE] = 1$ .
  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:  $TCDn\_SADDR = 0x1000$ ,  $TCDn\_DADDR = 0x2000$ ,  $TCDn\_CITER = 2$  ( $TCDn\_BITER$ ).

15. eDMA engine writes:  $TCDn\_CSR[ACTIVE] = 0$ ,  $TCDn\_CSR[DONE] = 1$ ,  $INT[n] = 1$ .
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

### 22.5.4.3 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 ( $0x1234567x$ ) 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 22-106. Modulo example**

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

## 22.5.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

### 22.5.5.1 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  $TCDn\_CITER$  field and test for a change. Another method may be extracted from the sequence shown below. The second method is

to test the `TCDn_CSR[START]` bit and the `TCDn_CSR[ACTIVE]` bit. The minor-loop-complete condition is indicated by both bits reading zero after the `TCDn_CSR[START]` was set. Polling the `TCDn_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	TCDn_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 `TCDn_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	TCDn_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 `TCDn_CSR[DONE]` bit.

The `TCDn_CSR[START]` bit is cleared automatically when the channel begins execution regardless of how the channel activates.

### 22.5.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true `TCDn_SADDR`, `TCDn_DADDR`, and `TCDn_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.

### 22.5.5.3 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 `TCDn_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 `TCDn_CSR[ACTIVE]` bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

### 22.5.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the `TCDn_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 `TCDn_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 ( $TCDn\_CITER[E\_LINK] = 1$ ), the  $TCDn\_CITER[CITER]$  field uses a nine bit vector to form the current iteration count. When minor loop linking is disabled ( $TCDn\_CITER[E\_LINK] = 0$ ), the  $TCDn\_CITER[CITER]$  field uses a 15-bit vector to form the current iteration count. The bits associated with the  $TCDn\_CITER[LINKCH]$  field are concatenated onto the CITER value to increase the range of the CITER.

### Note

The  $TCDn\_CITER[E\_LINK]$  bit and the  $TCDn\_BITER[E\_LINK]$  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 22-107. 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

## 22.5.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

### 22.5.7.1 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.

### 22.5.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e\_link bit during channel execution. 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.



### 22.5.7.3 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.

#### 22.5.7.3.1 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 identification (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 offset 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).

### **22.5.7.3.2 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 identification (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 offset value.

2. Write the TCD.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 23

## External Watchdog Monitor (EWM)

### 23.1 Introduction

#### NOTE

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

The watchdog is generally used to monitor the flow and execution of embedded software within an MCU. The watchdog consists of a counter that if allowed to overflow, forces an internal reset (asynchronous) to all on-chip peripherals and optionally assert the  $\overline{\text{RESET}}$  pin to reset external devices/circuits. The overflow of the watchdog counter must not occur if the software code works well and services the watchdog to re-start the actual counter.

For safety, a redundant watchdog system, External Watchdog Monitor (EWM), is designed to monitor external circuits, as well as the MCU software flow. This provides a back-up mechanism to the internal watchdog that resets the MCU's CPU and peripherals.

The EWM differs from the internal watchdog in that it does not reset the MCU's CPU and peripherals. The EWM if allowed to time-out, provides an independent  $\overline{\text{EWM\_out}}$  pin that when asserted resets or places an external circuit into a safe mode. The CPU resets the EWM counter that is logically ANDed with an external digital input pin. This pin allows an external circuit to influence the reset\_out signal.

#### 23.1.1 Features

Features of EWM module include:

- Independent LPO clock source
- Programmable time-out period specified in terms of number of EWM LPO clock cycles.

- Windowed refresh option
  - Provides robust check that program flow is faster than expected.
  - Programmable window.
  - Refresh outside window leads to assertion of  $\overline{\text{EWM\_out}}$ .
- Robust refresh mechanism
  - Write values of 0xB4 and 0x2C to EWM Refresh Register within 15 ( $\text{EWM\_service\_time}$ ) peripheral bus clock cycles.
- One output port,  $\overline{\text{EWM\_out}}$ , when asserted is used to reset or place the external circuit into safe mode.
- One Input port,  $\text{EWM\_in}$ , allows an external circuit to control the  $\overline{\text{EWM\_out}}$  signal.

### 23.1.2 Modes of Operation

This section describes the module's operating modes.

#### 23.1.2.1 Stop Mode

When the EWM is in stop mode, the CPU services to the EWM cannot occur. On entry to stop mode, the EWM's counter freezes.

There are two possible ways to exit from Stop mode:

- On exit from stop mode through a reset, the EWM remains disabled.
- On exit from stop mode by an interrupt, the EWM is re-enabled, and the counter continues to be clocked from the same value prior to entry to stop mode.

Note the following if the EWM enters the stop mode during CPU service mechanism: At the exit from stop mode by an interrupt, refresh mechanism state machine starts from the previous state which means, if first service command is written correctly and EWM enters the stop mode immediately, the next command has to be written within the next 15 ( $\text{EWM\_service\_time}$ ) peripheral bus clocks after exiting from stop mode. User must mask all interrupts prior to executing EWM service instructions.

### 23.1.2.2 Wait Mode

The EWM module treats the stop and wait modes as the same. EWM functionality remains the same in both of these modes.

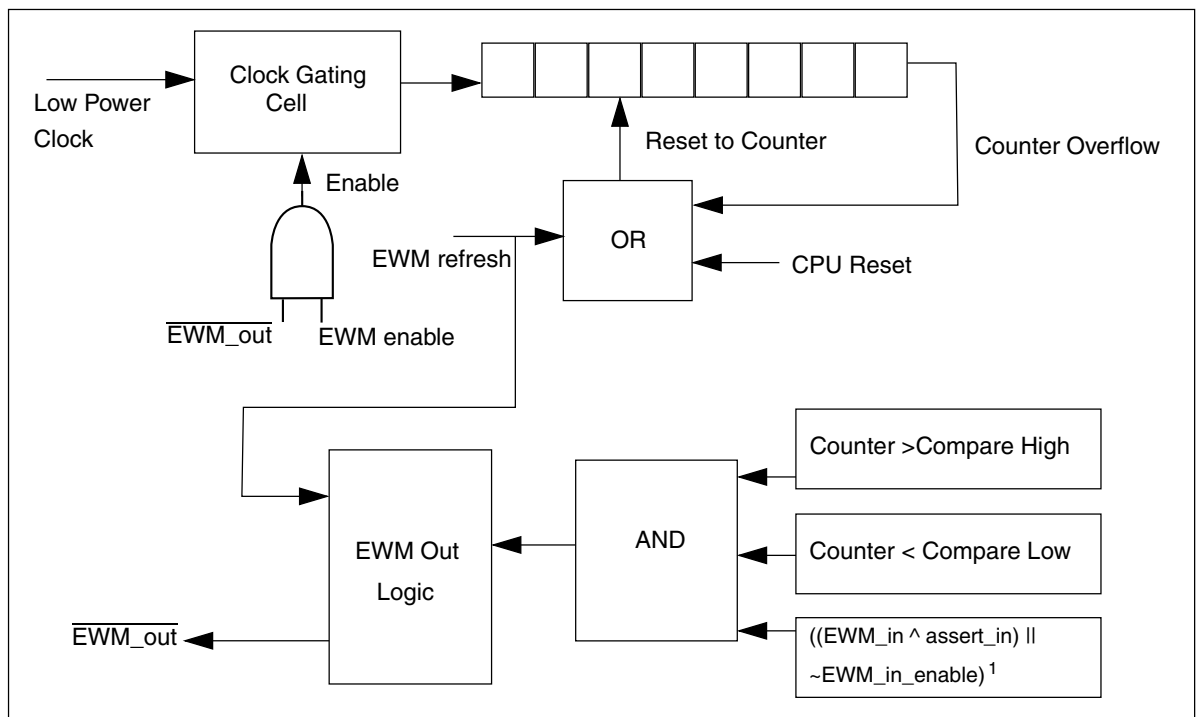
### 23.1.2.3 Debug Mode

Entry to debug mode has no effect on the EWM.

- If the EWM is enabled prior to entry of debug mode, it remains enabled.
- If the EWM is disabled prior to entry of debug mode, it remains disabled.

## 23.1.3 Block Diagram

This figure shows the EWM block diagram.



<sup>1</sup> Compare High > Counter > Compare Low

**Figure 23-1. EWM Block Diagram**

## 23.2 EWM Signal Descriptions

The EWM has two external signals, as shown in the following table.

**Table 23-1. 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

## 23.3 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	<a href="#">23.3.1/440</a>
4006_1001	Service Register (EWM_SERV)	8	W (always reads 0)	00h	<a href="#">23.3.2/441</a>
4006_1002	Compare Low Register (EWM_CMPL)	8	R/W	00h	<a href="#">23.3.3/441</a>
4006_1003	Compare High Register (EWM_CMPH)	8	R/W	FFh	<a href="#">23.3.4/442</a>
4006_1005	Clock Prescaler Register (EWM_CLKPRESCALER)	8	R/W	00h	<a href="#">23.3.5/443</a>

### 23.3.1 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	0				0	0	0	0
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 ASSIN bit inverts the assert state to a logic one.
0 EWMEN	EWM enable. This bit when set, enables the EWM module. This resets the EWM counter to zero and deasserts the $\overline{\text{EWM\_out}}$ signal. Clearing EWMEN bit disables the EWM, and therefore it cannot be enabled until a reset occurs, due to the write-once nature of this bit.

### 23.3.2 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
7–0 SERVICE	The EWM service mechanism requires the CPU to write two values to the SERV register: a first data byte of 0xB4, followed by a second data byte of 0x2C. The EWM service is illegal if either of the following conditions is true. <ul style="list-style-type: none"> <li>The first or second data byte is not written correctly.</li> <li>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 <i>EWM_service_time</i>.</li> </ul>

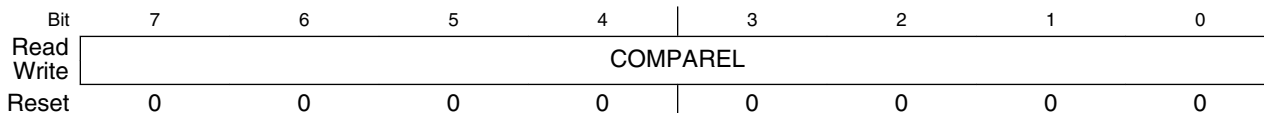
### 23.3.3 Compare Low Register (EWM\_CMPL)

The CMPL register is reset to zero after a CPU reset. This provides no minimum time for the CPU to service the EWM counter.

**NOTE**

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

Address: 4006\_1000h base + 2h offset = 4006\_1002h



**EWM\_CMPLE field descriptions**

Field	Description
7-0 COMPAREL	To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) minimum service time is required.

**23.3.4 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 service the EWM counter.

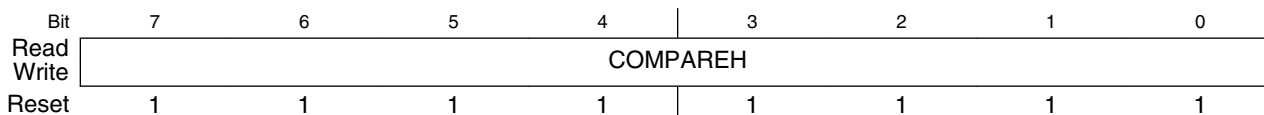
**NOTE**

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

**NOTE**

The valid values for CMPH are up to 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



**EWM\_CMPH field descriptions**

Field	Description
7-0 COMPAREH	To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) maximum service time is required.

### 23.3.5 Clock Prescaler Register (EWM\_CLKPRESCALER)

This CLKPRESCALER register is reset to 0x00 after a CPU reset.

#### NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

#### NOTE

Write the required prescaler value before enabling the EWM.

#### NOTE

The implementation of this register is chip-specific. See the Chip Configuration details.

Address: 4006\_1000h base + 5h offset = 4006\_1005h

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

#### EWM\_CLKPRESCALER field descriptions

Field	Description
7-0 CLK_DIV	Selected low power source for running the EWM counter can be prescaled as below. <ul style="list-style-type: none"> <li>Prescaled clock frequency = low power clock source frequency / ( 1 + CLK_DIV )</li> </ul>

## 23.4 Functional Description

The following sections describe functional details of the EWM module.

### 23.4.1 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. For example, the  $\overline{\text{EWM\_out}}$  could be connected to the high voltage transistors circuits that control an AC motor in a large appliance.

The  $\overline{\text{EWM\_out}}$  signal remains deasserted when the EWM is being regularly serviced by the CPU within the programmable service window, indicating that the application code is executed as expected.

The  $\overline{\text{EWM\_out}}$  signal is asserted in any of the following conditions:

- Servicing the EWM when the counter value is less than CMPL value.
- If the EWM counter value reaches the CMPH value, and no EWM service has occurred.
- Servicing the EWM when the counter value is more than CMPL and less than CMPH values and EWM\_in signal is asserted.
- If functionality of EWM\_in pin is enabled and EWM\_in pin is asserted while servicing the EWM.
- After any reset (by the virtue of the external pull-down mechanism on the  $\overline{\text{EWM\_out}}$  pin)

On a normal reset, the  $\overline{\text{EWM\_out}}$  is asserted. To deassert the  $\overline{\text{EWM\_out}}$ , set EWMEN bit in the CTRL register to enable the EWM.

If the  $\overline{\text{EWM\_out}}$  signal shares its pad with a digital I/O pin, on reset this actual pad defers to being an input signal. It takes the  $\overline{\text{EWM\_out}}$  output condition only after you enable the EWM by the EWMEN bit in the CTRL register.

When the  $\overline{\text{EWM\_out}}$  pin is asserted, it can only be deasserted by forcing a MCU reset.

### Note

$\overline{\text{EWM\_out}}$  pad must be in pull down state when EWM functionality is used and when EWM is under Reset.

## 23.4.2 The EWM\_in Signal

The EWM\_in is a digital input signal that allows an external circuit to control 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 this circuit's behavior, it 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 servicing the EWM. This ensures that the  $\overline{\text{EWM\_out}}$  stays in the deasserted state; otherwise, the  $\overline{\text{EWM\_out}}$  pin is asserted.

## Note

You must update the CMPH and CMPL registers prior to enabling the EWM. After enabling the EWM, the counter resets to zero, therefore providing a reasonable time after a power-on reset for the external monitoring circuit to stabilize and ensure that the EWM\_in pin is deasserted.

### 23.4.3 EWM Counter

It is an 8-bit ripple counter fed from a clock source that is independent of the peripheral bus clock source. As the preferred time-out is between 1 ms and 100 ms the actual clock source should be in the kHz range.

The counter is reset to zero, after a CPU reset, or a EWM refresh cycle. The counter value is not accessible to the CPU.

### 23.4.4 EWM Compare Registers

The compare registers CMPL and CMPH are write-once after a CPU reset and cannot be modified until another CPU reset occurs.

The EWM compare registers are used to create a service window, which is used by the CPU to service/refresh the EWM module.

- If the CPU services the EWM when the counter value lies between CMPL value and CMPH value, the counter is reset to zero. This is a legal service operation.
- If the CPU executes a EWM service/refresh action outside the legal service window, EWM\_out is asserted.

It is illegal to program CMPL and CMPH with same value. In this case, as soon as counter reaches (CMPL + 1), EWM\_out is asserted.

### 23.4.5 EWM Refresh Mechanism

Other than the initial configuration of the EWM, the CPU can only access the EWM by the EWM Service Register. The CPU must access the EWM service register with correct write of unique data within the windowed time frame as determined by the CMPL and CMPH registers. Therefore, three possible conditions can occur:

**Table 23-8. EWM Refresh Mechanisms**

Condition	Mechanism
A unique EWM service occurs when $CMPL < Counter < CMPH$ .	The software behaves as expected and the counter of the EWM is reset to zero, and $\overline{EWM\_out}$ pin remains in the deasserted state. <b>Note:</b> $EWM\_in$ pin is also assumed to be in the deasserted state.
A unique EWM service occurs when $Counter < CMPL$	The software services the EWM and therefore resets the counter to zero and asserts the $\overline{EWM\_out}$ pin (irrespective of the $EWM\_in$ pin). The $\overline{EWM\_out}$ pin is expected to gate critical safety circuits.
Counter value reaches $CMPH$ prior to a unique EWM service	The counter value reaches the $CMPH$ value and no service of the EWM resets the counter to zero and assert the $\overline{EWM\_out}$ pin (irrespective of the $EWM\_in$ pin). The $\overline{EWM\_out}$ pin is expected to gate critical safety circuits.

Any illegal service on EWM has no effect on  $\overline{EWM\_out}$ .

### 23.4.6 EWM Interrupt

When  $\overline{EWM\_out}$  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  $\overline{EWM\_out}$ . The  $\overline{EWM\_out}$  signal can be deasserted only by forcing a system reset.

### 23.4.7 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 24

## Watchdog Timer (WDOG)

### 24.1 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.

### 24.2 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.

## Features

- 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.

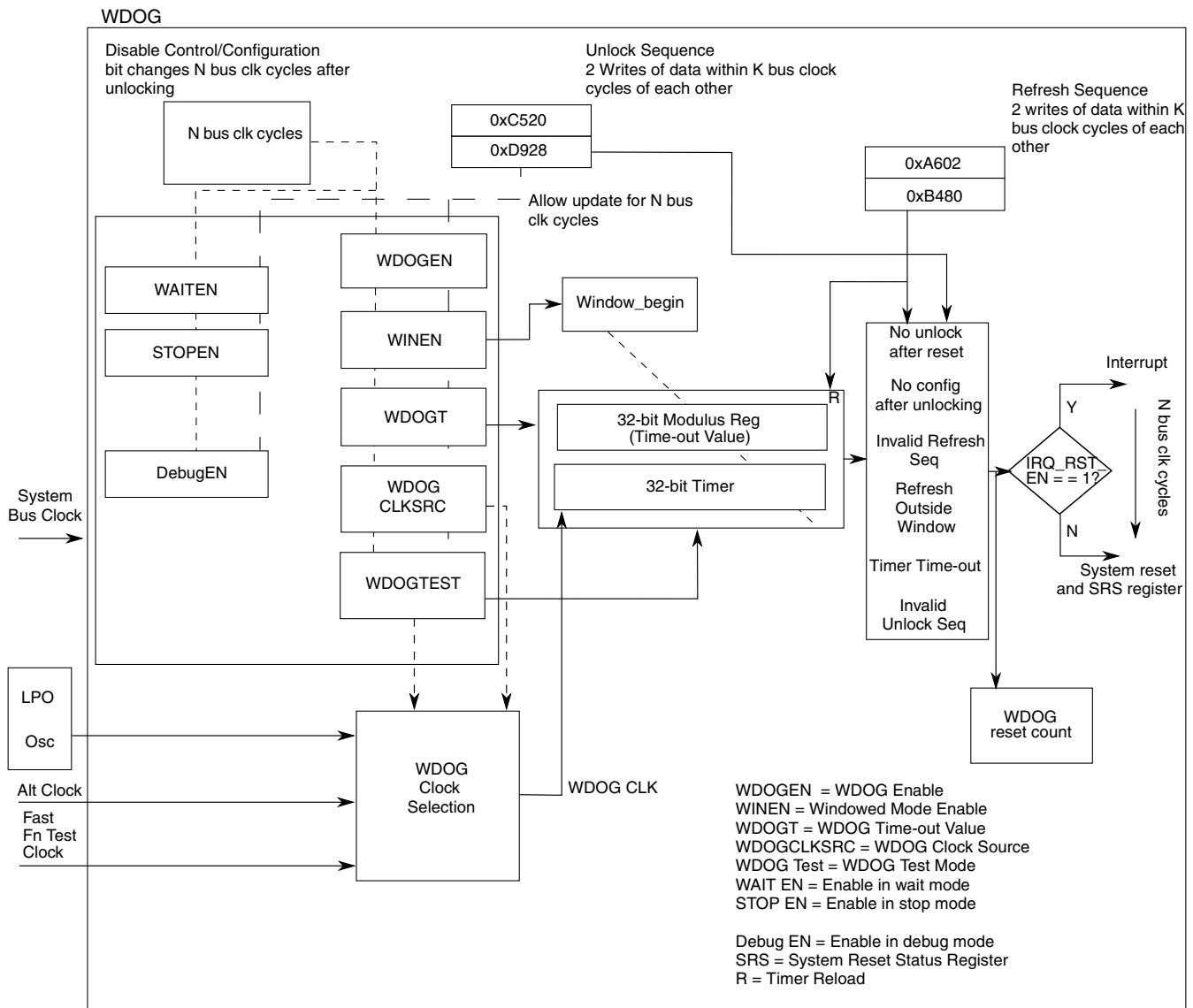
### NOTE

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.



## 24.3 Functional overview



**Figure 24-1. 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.

### **24.3.1 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.

## 24.3.2 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  $2 \times \text{WCT} + 20$  bus clock cycles. Therefore, it must be ensured that the time-out value for the watchdog is always greater than  $2 \times \text{WCT} + 20$  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.

### 24.3.3 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.

### 24.3.4 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.

### 24.3.5 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.

### 24.3.6 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 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 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 mode, the WDOG timer restarts and the WDOG has to be unlocked and configured within WCT.

## 24.4 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

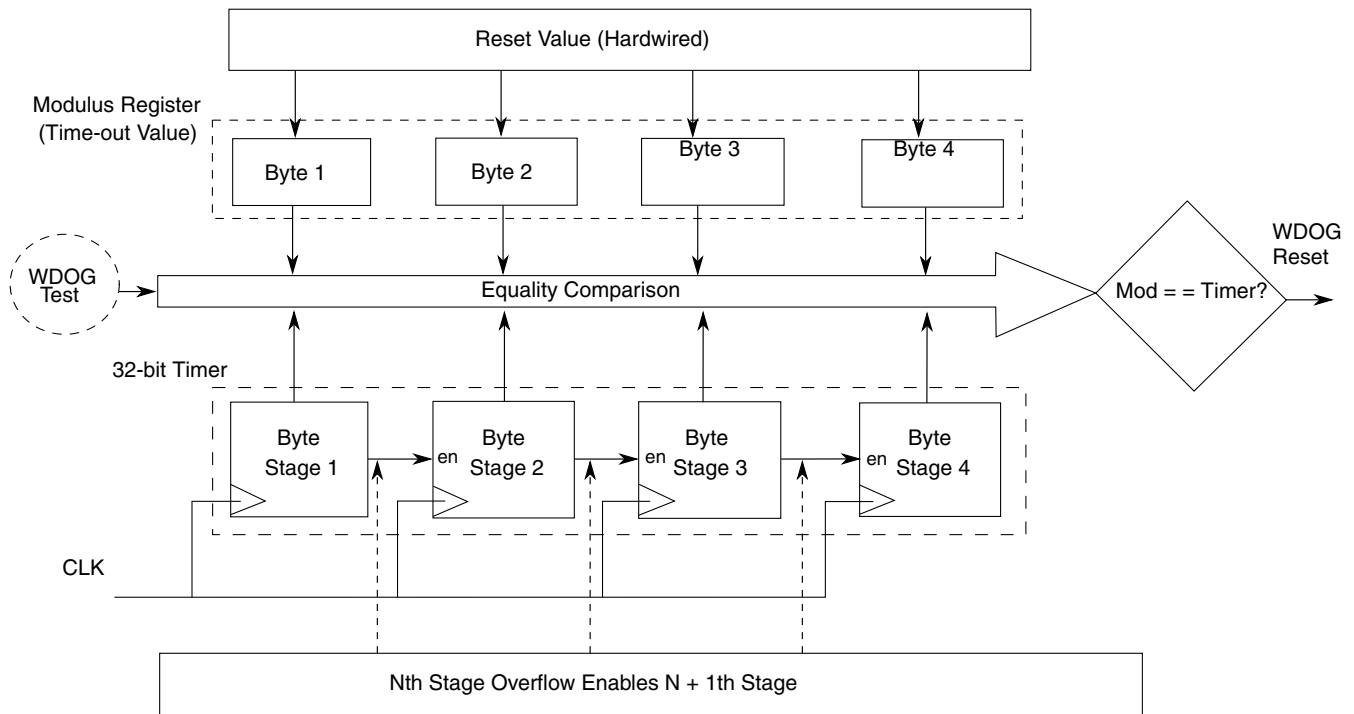
After emerging from a reset due to a watchdog test, unlock and configure the watchdog. The refresh and unlock operations and interrupt are not automatically disabled in the test mode.

## 24.4.1 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.

## 24.4.2 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 24-2. 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  $N + 1$ th 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  $N$ th stage runs and compares with the  $N$ th 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.

## 24.5 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.

## 24.6 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.

## 24.7 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	<a href="#">24.7.1/457</a>
4005_2002	Watchdog Status and Control Register Low (WDOG_STCTRLLL)	16	R/W	0001h	<a href="#">24.7.2/459</a>
4005_2004	Watchdog Time-out Value Register High (WDOG_TOVALH)	16	R/W	004Ch	<a href="#">24.7.3/459</a>
4005_2006	Watchdog Time-out Value Register Low (WDOG_TOVALL)	16	R/W	4B4Ch	<a href="#">24.7.4/460</a>
4005_2008	Watchdog Window Register High (WDOG_WINH)	16	R/W	0000h	<a href="#">24.7.5/460</a>
4005_200A	Watchdog Window Register Low (WDOG_WINL)	16	R/W	0010h	<a href="#">24.7.6/461</a>
4005_200C	Watchdog Refresh register (WDOG_REFRESH)	16	R/W	B480h	<a href="#">24.7.7/461</a>
4005_200E	Watchdog Unlock register (WDOG_UNLOCK)	16	R/W	D928h	<a href="#">24.7.8/461</a>
4005_2010	Watchdog Timer Output Register High (WDOG_TMROUTH)	16	R/W	0000h	<a href="#">24.7.9/462</a>
4005_2012	Watchdog Timer Output Register Low (WDOG_TMROUTL)	16	R/W	0000h	<a href="#">24.7.10/462</a>
4005_2014	Watchdog Reset Count register (WDOG_RSTCNT)	16	R/W	0000h	<a href="#">24.7.11/463</a>
4005_2016	Watchdog Prescaler register (WDOG_PRESC)	16	R/W	0400h	<a href="#">24.7.12/463</a>

## 24.7.1 Watchdog Status and Control Register High (WDOG\_STCTRLH)

Address: 4005\_2000h base + 0h offset = 4005\_2000h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	DISTESTWDOG	BYTESEL[1:0]		TESTSEL	TESTWDOG	0	Reserved	WAITEN	STOPEN	DBGEN	ALLOWUPDATE	WINEN	IRQRSTEN	CLKSRC	WDOGEN
Write																
Reset	0	0	0	0	0	0	0	1	1	1	0	1	0	0	1	1

### 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

Table continues on the next page...

## WDOG\_STCTRLH field descriptions (continued)

Field	Description
	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.  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.

Table continues on the next page...

## WDOG\_STCTRLH field descriptions (continued)

Field	Description
0	WDOG is disabled.
1	WDOG is enabled.

## 24.7.2 Watchdog Status and Control Register Low (WDOG\_STCTRL)

Address: 4005\_2000h base + 2h offset = 4005\_2002h

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

## WDOG\_STCTRL 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.
14–0 Reserved	This field is reserved. <b>NOTE:</b> Do not modify this field value.

## 24.7.3 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	TOVALHIGH															
Write																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0

## WDOG\_TOVALH field descriptions

Field	Description
15–0 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.

### 24.7.4 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	TOVALLOW															
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
15–0 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.

### 24.7.5 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	WINHIGH															
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
15–0 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.

## 24.7.6 Watchdog Window Register Low (WDOG\_WINL)

### NOTE

You must set the Window Register value lower than the Timeout 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	WINLOW															
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
15–0 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.

## 24.7.7 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	WDOGREFRESH															
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
15–0 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.

## 24.7.8 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	WDOGUNLOCK															
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
15–0 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 IRQSTEN is set, it interrupts and then resets the system. The unlock sequence is effective only if ALLOWUPDATE is set.

### 24.7.9 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	TIMEROUTHIGH																
Write																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

#### WDOG\_TMROUTH field descriptions

Field	Description
15–0 TIMEROUTHIGH	Shows the value of the upper 16 bits of the watchdog timer.

### 24.7.10 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	TIMEROUTLOW																
Write																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

#### WDOG\_TMROUTL field descriptions

Field	Description
15–0 TIMEROUTLOW	Shows the value of the lower 16 bits of the watchdog timer.

### 24.7.11 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	RSTCNT															
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
15–0 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.

### 24.7.12 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	PRESCVAL															
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.
7–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

## 24.8 Watchdog operation with 8-bit access

### 24.8.1 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.

## 24.8.2 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 24-14. Refresh for 8-bit access**

	WDOG_REFRESH[15:8]	WDOG_REFRESH[7:0]	Sequence value1 or value2 match	Mismatch exception
<b>Current Value</b>	0xB4	0x80	Value2 match	No
<b>Write 1</b>	0xB4	0x02	No match	No
<b>Write 2</b>	0xA6	0x02	Value1 match	No
<b>Write 3</b>	0xB4	0x02	No match	No
<b>Write 4</b>	0xB4	0x80	Value2 match. Sequence complete.	No
<b>Write 5</b>	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](#).

## 24.9 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  $\sim 2$  clock A cycles plus  $\sim 2$  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.

## Restrictions on watchdog operation

- 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  $2 \times \text{WCT time} + 20$  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 25

## Multipurpose Clock Generator (MCG)

### 25.1 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 conjunction with a crystal oscillator, which allows an external crystal, ceramic resonator, or another external clock source to produce the external reference clock.

#### 25.1.1 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 from the Real Time Counter (RTC):
  - Can only 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.

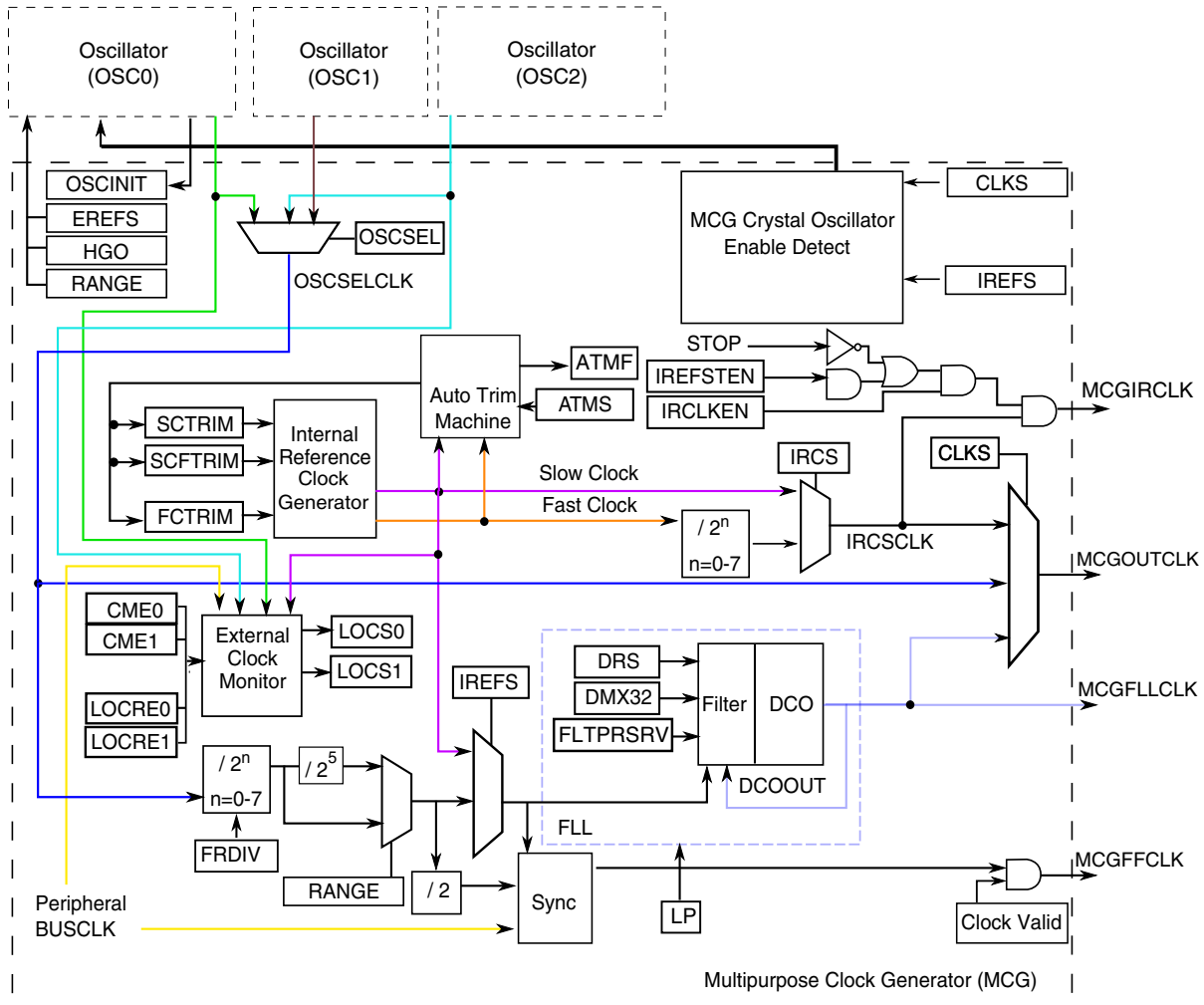


Figure 25-1. Multipurpose Clock Generator (MCG) block diagram

**NOTE**

Refer to the chip configuration chapter to identify the oscillator used in this MCU.

## 25.1.2 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](#).

## 25.2 External Signal Description

There are no MCG signals that connect off chip.

## 25.3 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	<a href="#">25.3.1/470</a>
4006_4001	MCG Control 2 Register (MCG_C2)	8	R/W	80h	<a href="#">25.3.2/472</a>
4006_4002	MCG Control 3 Register (MCG_C3)	8	R/W	Undefined	<a href="#">25.3.3/473</a>
4006_4003	MCG Control 4 Register (MCG_C4)	8	R/W	Undefined	<a href="#">25.3.4/474</a>
4006_4005	MCG Control 6 Register (MCG_C6)	8	R/W	00h	<a href="#">25.3.5/475</a>
4006_4006	MCG Status Register (MCG_S)	8	R	10h	<a href="#">25.3.6/476</a>
4006_4008	MCG Status and Control Register (MCG_SC)	8	R/W	02h	<a href="#">25.3.7/477</a>
4006_400A	MCG Auto Trim Compare Value High Register (MCG_ATCVH)	8	R/W	00h	<a href="#">25.3.8/478</a>
4006_400B	MCG Auto Trim Compare Value Low Register (MCG_ATCVL)	8	R/W	00h	<a href="#">25.3.9/478</a>
4006_400C	MCG Control 7 Register (MCG_C7)	8	R/W	00h	<a href="#">25.3.10/479</a>
4006_400D	MCG Control 8 Register (MCG_C8)	8	R/W	<a href="#">See section</a>	<a href="#">25.3.11/479</a>

### 25.3.1 MCG Control 1 Register (MCG\_C1)

Address: 4006\_4000h base + 0h offset = 4006\_4000h

Bit	7	6	5	4	3	2	1	0
Read	CLKS		FRDIV			IREFS	IRCLKEN	IREFSTEN
Write								
Reset	0	0	0	0	0	1	0	0

## MCG\_C1 field descriptions

Field	Description
7–6 CLKS	<p>Clock Source Select</p> <p>Selects the clock source for MCGOUTCLK .</p> <p>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.</p>
5–3 FRDIV	<p>FLL External Reference Divider</p> <p>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).</p> <p>000 If RANGE = 0 or OSCSEL=1 , Divide Factor is 1; for all other RANGE values, Divide Factor is 32.  001 If RANGE = 0 or OSCSEL=1 , Divide Factor is 2; for all other RANGE values, Divide Factor is 64.  010 If RANGE = 0 or OSCSEL=1 , Divide Factor is 4; for all other RANGE values, Divide Factor is 128.  011 If RANGE = 0 or OSCSEL=1 , Divide Factor is 8; for all other RANGE values, Divide Factor is 256.  100 If RANGE = 0 or OSCSEL=1 , Divide Factor is 16; for all other RANGE values, Divide Factor is 512.  101 If RANGE = 0 or OSCSEL=1 , Divide Factor is 32; for all other RANGE values, Divide Factor is 1024.  110 If RANGE = 0 or OSCSEL=1 , Divide Factor is 64; for all other RANGE values, Divide Factor is 1280 .  111 If RANGE = 0 or OSCSEL=1 , Divide Factor is 128; for all other RANGE values, Divide Factor is 1536 .</p>
2 IREFS	<p>Internal Reference Select</p> <p>Selects the reference clock source for the FLL.</p> <p>0 External reference clock is selected.  1 The slow internal reference clock is selected.</p>
1 IRCLKEN	<p>Internal Reference Clock Enable</p> <p>Enables the internal reference clock for use as MCGIRCLK.</p> <p>0 MCGIRCLK inactive.  1 MCGIRCLK active.</p>
0 IREFSTEN	<p>Internal Reference Stop Enable</p> <p>Controls whether or not the internal reference clock remains enabled when the MCG enters Stop mode.</p> <p>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.</p>

## 25.3.2 MCG Control 2 Register (MCG\_C2)

Address: 4006\_4000h base + 1h offset = 4006\_4001h

Bit	7	6	5	4	3	2	1	0
Read	LOCRE0	FCFTRIM	RANGE		HGO	EREFS	LP	IRCS
Write								
Reset	1	0	0	0	0	0	0	0

### MCG\_C2 field descriptions

Field	Description
7 LOCRE0	<p>Loss of Clock Reset Enable</p> <p>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.</p> <p>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.</p>
6 FCFTRIM	<p>Fast Internal Reference Clock Fine Trim</p> <p>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.</p>
5-4 RANGE	<p>Frequency Range Select</p> <p>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.</p> <p>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 .</p>
3 HGO	<p>High Gain Oscillator Select</p> <p>Controls the crystal oscillator mode of operation. See the Oscillator (OSC) chapter for more details.</p> <p>0 Configure crystal oscillator for low-power operation. 1 Configure crystal oscillator for high-gain operation.</p>
2 EREFS	<p>External Reference Select</p> <p>Selects the source for the external reference clock. See the Oscillator (OSC) chapter for more details.</p> <p>0 External reference clock requested. 1 Oscillator requested.</p>
1 LP	<p>Low Power Select</p> <p>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.</p> <p>0 FLL is not disabled in bypass modes. 1 FLL is disabled in bypass modes (lower power)</p>

Table continues on the next page...



**MCG\_C2 field descriptions (continued)**

Field	Description
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.

**25.3.3 MCG Control 3 Register (MCG\_C3)**

Address: 4006\_4000h base + 2h offset = 4006\_4002h

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

\* Notes:

- x = Undefined at reset.

**MCG\_C3 field descriptions**

Field	Description
7–0 SCTRIM	Slow Internal Reference Clock Trim Setting  SCTRIM <sup>1</sup> 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.

### 25.3.4 MCG Control 4 Register (MCG\_C4)

#### NOTE

Reset values for DRST and DMX32 bits are 0.

Address: 4006\_4000h base + 3h offset = 4006\_4003h

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

\* Notes:

- x = Undefined at reset.
- A value for FCTRIM is loaded during reset from a factory programmed location. x = Undefined at reset.

#### MCG\_C4 field descriptions

Field	Description																																									
7 DMX32	<p>DCO Maximum Frequency with 32.768 kHz Reference</p> <p>The DMX32 bit controls whether the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference.</p> <p>The following table identifies settings for the DCO frequency range.</p> <p><b>NOTE:</b> The system clocks derived from this source should not exceed their specified maximums.</p> <table border="1"> <thead> <tr> <th>DRST_DRS</th> <th>DMX32</th> <th>Reference Range</th> <th>FLL Factor</th> <th>DCO Range</th> </tr> </thead> <tbody> <tr> <td rowspan="2">00</td> <td>0</td> <td>31.25–39.0625 kHz</td> <td>640</td> <td>20–25 MHz</td> </tr> <tr> <td>1</td> <td>32.768 kHz</td> <td>732</td> <td>24 MHz</td> </tr> <tr> <td rowspan="2">01</td> <td>0</td> <td>31.25–39.0625 kHz</td> <td>1280</td> <td>40–50 MHz</td> </tr> <tr> <td>1</td> <td>32.768 kHz</td> <td>1464</td> <td>48 MHz</td> </tr> <tr> <td rowspan="2">10</td> <td>0</td> <td>31.25–39.0625 kHz</td> <td>1920</td> <td>60–75 MHz</td> </tr> <tr> <td>1</td> <td>32.768 kHz</td> <td>2197</td> <td>72 MHz</td> </tr> <tr> <td rowspan="2">11</td> <td>0</td> <td>31.25–39.0625 kHz</td> <td>2560</td> <td>80–100 MHz</td> </tr> <tr> <td>1</td> <td>32.768 kHz</td> <td>2929</td> <td>96 MHz</td> </tr> </tbody> </table> <p>0 DCO has a default range of 25%. 1 DCO is fine-tuned for maximum frequency with 32.768 kHz reference.</p>	DRST_DRS	DMX32	Reference Range	FLL Factor	DCO Range	00	0	31.25–39.0625 kHz	640	20–25 MHz	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
DRST_DRS	DMX32	Reference Range	FLL Factor	DCO Range																																						
00	0	31.25–39.0625 kHz	640	20–25 MHz																																						
	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																																						
6–5 DRST_DRS	<p>DCO Range Select</p> <p>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.</p> <p>00 Encoding 0 — Low range (reset default).</p>																																									

Table continues on the next page...

## MCG\_C4 field descriptions (continued)

Field	Description
	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 <sup>1</sup> 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 <sup>2</sup> 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 .

## 25.3.5 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		CME			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 CME	Clock Monitor Enable  Determines if a reset request is made following a loss of external clock indication. The CME 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 CME bit is set to a logic 1, the value of the RANGE bits in the C2 register should not be changed. CME 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. CME 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 a reset request on loss of external clock.
4–0 Reserved	Reserved

Table continues on the next page...

## MCG\_C6 field descriptions (continued)

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

## 25.3.6 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 CLKST 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 (IRCSCCLK). The IRCST bit does not update immediately after a write to the IRCST 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).

## 25.3.7 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								
Write	ATME	ATMS	ATMF	FLTPRSRV		FCDIV		LOCS0
Reset	0	0	0	0	0	0	1	0

### MCG\_SC field descriptions

Field	Description
7 ATME	<p>Automatic Trim Machine Enable</p> <p>Enables the Auto Trim Machine to start automatically trimming the selected Internal Reference Clock.</p> <p><b>NOTE:</b> ATME deasserts after the Auto Trim Machine has completed trimming all trim bits of the IRCS clock selected by the ATMS bit.</p> <p>Writing to C1, C3, C4, and SC registers or entering Stop mode aborts the auto trim operation and clears this bit.</p> <p>0 Auto Trim Machine disabled. 1 Auto Trim Machine enabled.</p>
6 ATMS	<p>Automatic Trim Machine Select</p> <p>Selects the IRCS clock for Auto Trim Test.</p> <p>0 32 kHz Internal Reference Clock selected. 1 4 MHz Internal Reference Clock selected.</p>
5 ATMF	<p>Automatic Trim Machine Fail Flag</p> <p>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.</p> <p>0 Automatic Trim Machine completed normally. 1 Automatic Trim Machine failed.</p>
4 FLTPRSRV	<p>FLL Filter Preserve Enable</p> <p>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.)</p> <p>0 FLL filter and FLL frequency will reset on changes to correct clock mode. 1 FLL filter and FLL frequency retain their previous values during new clock mode change.</p>
3-1 FCDIV	<p>Fast Clock Internal Reference Divider</p> <p>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).</p> <p>000 Divide Factor is 1 001 Divide Factor is 2.</p>

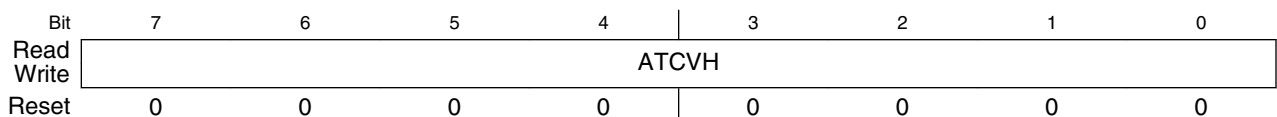
Table continues on the next page...

**MCG\_SC field descriptions (continued)**

Field	Description
	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.

**25.3.8 MCG Auto Trim Compare Value High Register (MCG\_ATCVH)**

Address: 4006\_4000h base + Ah offset = 4006\_400Ah

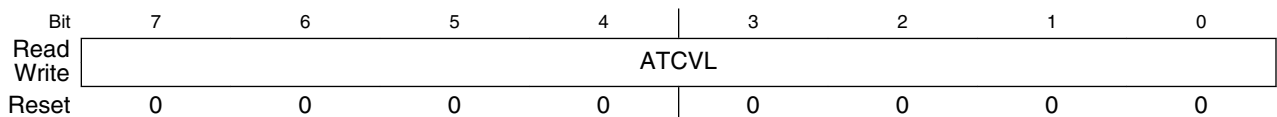


**MCG\_ATCVH field descriptions**

Field	Description
7-0 ATCVH	ATM Compare Value High  Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion.

**25.3.9 MCG Auto Trim Compare Value Low Register (MCG\_ATCVL)**

Address: 4006\_4000h base + Bh offset = 4006\_400Bh



**MCG\_ATCVL field descriptions**

Field	Description
7-0 ATCVL	ATM Compare Value Low  Values are used by Auto Trim Machine to compare and adjust Internal Reference trim values during ATM SAR conversion.

### 25.3.10 MCG Control 7 Register (MCG\_C7)

Address: 4006\_4000h base + Ch offset = 4006\_400Ch

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

#### MCG\_C7 field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–2 Reserved	Reserved  This field is reserved. This read-only field is reserved and always has the value 0.
1–0 OSCSEL	MCG OSC Clock Select  Selects the MCG FLL external reference clock  <b>NOTE:</b> The OSCSEL field can't be changed during MCG modes (like PBE), when external clock is serving as the clock source for MCG.  00 Selects Oscillator (OSCCLK0). 01 Selects 32 kHz RTC Oscillator. 10 Selects Oscillator (OSCCLK1). 11 RESERVED

### 25.3.11 MCG Control 8 Register (MCG\_C8)

Address: 4006\_4000h base + Dh offset = 4006\_400Dh

Bit	7	6	5	4	3	2	1	0
Read	LOCRE1	0	CME1	0			LOCS1	
Write								
Reset	1	0	0	0	0	0	0	0

#### MCG\_C8 field descriptions

Field	Description
7 LOCRE1	Loss of Clock Reset Enable  Determines if a interrupt or a reset request is made following a loss of RTC external reference clock. The LOCRE1 only has an affect when CME1 is set.  0 Interrupt request is generated on a loss of RTC external reference clock. 1 Generate a reset request on a loss of RTC external reference clock

Table continues on the next page...

**MCG\_C8 field descriptions (continued)**

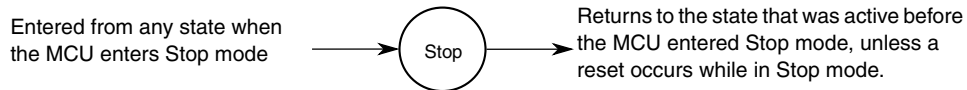
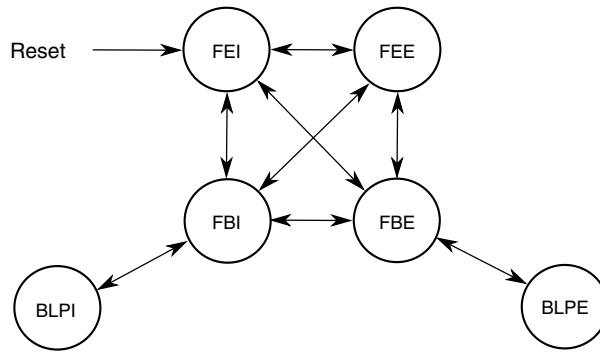
Field	Description
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 CME1	<p>Clock Monitor Enable1</p> <p>Enables the loss of clock monitoring circuit for the output of the RTC external reference clock. The LOCRE1 bit will determine whether an interrupt or a reset request is generated following a loss of RTC clock indication. The CME1 bit should be set to a logic 1 when the MCG is in an operational mode that uses the RTC as its external reference clock or if the RTC is operational. CME1 bit must be set to a logic 0 before the MCG enters any Stop mode. Otherwise, a reset request may occur when in Stop mode. CME1 should also be set to a logic 0 before entering VLPR or VLPW power modes.</p> <p>0 External clock monitor is disabled for RTC clock. 1 External clock monitor is enabled for RTC clock.</p>
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 LOCS1	<p>RTC Loss of Clock Status</p> <p>This bit indicates when a loss of clock has occurred. This bit is cleared by writing a logic 1 to it when set.</p> <p>0 Loss of RTC has not occur. 1 Loss of RTC has occur</p>

## 25.4 Functional description

### 25.4.1 MCG mode state diagram

The seven states of the MCG are shown in the following figure and are described in [Table 25-13](#). The arrows indicate the permitted MCG mode transitions.





**Figure 25-13. MCG mode state diagram**

### 25.4.1.1 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 25-13](#).

**Table 25-13. MCG modes of operation**

Mode	Description
FLL Engaged Internal (FEI)	<p>FLL engaged internal (FEI) is the default mode of operation and is entered when all the following conditions occur:</p> <ul style="list-style-type: none"> <li>• 00 is written to C1[CLKS].</li> <li>• 1 is written to C1[IREFS].</li> </ul> <p>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.</p>

*Table continues on the next page...*

**Table 25-13. MCG modes of operation (continued)**

Mode	Description
FLL Engaged External (FEE)	<p>FLL engaged external (FEE) mode is entered when all the following conditions occur:</p> <ul style="list-style-type: none"> <li>• 00 is written to C1[CLKS].</li> <li>• 0 is written to C1[IREFS].</li> <li>• C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz</li> </ul> <p>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.</p>
FLL Bypassed Internal (FBI)	<p>FLL bypassed internal (FBI) mode is entered when all the following conditions occur:</p> <ul style="list-style-type: none"> <li>• 01 is written to C1[CLKS].</li> <li>• 1 is written to C1[IREFS].</li> <li>• 0 is written to C2[LP].</li> </ul> <p>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.</p>
FLL Bypassed External (FBE)	<p>FLL bypassed external (FBE) mode is entered when all the following conditions occur:</p> <ul style="list-style-type: none"> <li>• 10 is written to C1[CLKS].</li> <li>• 0 is written to C1[IREFS].</li> <li>• C1[FRDIV] must be written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.</li> <li>• 0 is written to C2[LP].</li> </ul> <p>In FBE mode, the MCGOUTCLK is derived from the OSCSEL 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.</p>
Bypassed Low Power Internal (BLPI) <sup>1</sup>	<p>Bypassed Low Power Internal (BLPI) mode is entered when all the following conditions occur:</p> <ul style="list-style-type: none"> <li>• 01 is written to C1[CLKS].</li> <li>• 1 is written to C1[IREFS].</li> <li>• 1 is written to C2[LP].</li> </ul> <p>In BLPI mode, MCGOUTCLK is derived from the internal reference clock. The FLL is disabled</p>

*Table continues on the next page...*

**Table 25-13. MCG modes of operation (continued)**

Mode	Description
Bypassed Low Power External (BLPE)	<p>Bypassed Low Power External (BLPE) mode is entered when all the following conditions occur:</p> <ul style="list-style-type: none"> <li>• 10 is written to C1[CLKS].</li> <li>• 0 is written to C1[IREFS].</li> <li>• 1 is written to C2[LP].</li> </ul> <p>In BLPE mode, MCGOUTCLK is derived from the OSCSEL external reference clock. The FLL is disabled</p>
Stop	<p>Entered whenever the MCU enters a Stop state. The power modes are chip specific. For power mode assignments, see the 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:</p> <p>MCGIRCLK is active in Normal Stop mode when all the following conditions become true:</p> <ul style="list-style-type: none"> <li>• C1[IRCLKEN] = 1</li> <li>• C1[IREFSTEN] = 1</li> </ul>

1. 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.

#### 25.4.1.2 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  $t_{\text{fll\_acquire}}$ .

## 25.4.2 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].

## 25.4.3 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.

### 25.4.3.1 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.

## 25.4.4 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]). For the case when C8[CME1]=1, a loss of clock is detected if the RTC external reference falls below a minimum frequency ( $f_{loc\_low}$ ).

### NOTE

All clock monitors must be disabled before entering these low-power modes: Stop, VLPS, VLPR, VLPW, LLS, 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.

## 25.4.5 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.

## 25.4.6 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:

$$\text{ATCV Expected Count Value} = 21 * (\text{Fe} / \text{Fr})$$

- 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.

$$\text{Expected Count Value} = (\text{Fe} / \text{Fr}) * 21 * (128)$$

## 25.5 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.

### 25.5.1 MCG module initialization sequence

The MCG comes out of reset configured for FEI mode.

The internal reference will stabilize in  $t_{\text{irefstb}}$  microseconds before the FLL can acquire lock. As soon as the internal reference is stable, the FLL will acquire lock in  $t_{\text{fll\_acquire}}$  milliseconds.

#### 25.5.1.1 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 25-13](#)). 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[EREFS0] 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.

### 25.5.2 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.

### 25.5.3 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[EREFS0]), 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 divide 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 25-14. MCGOUTCLK Frequency Calculation Options**

Clock Mode	$f_{\text{MCGOUTCLK}}^1$	Note
FEI (FLL engaged internal)	$(f_{\text{int}} * F)$	Typical $f_{\text{MCGOUTCLK}} = 21$ MHz immediately after reset.
FEE (FLL engaged external)	$(f_{\text{ext}} / \text{FLL\_R}) * F$	$f_{\text{ext}} / \text{FLL\_R}$ 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	

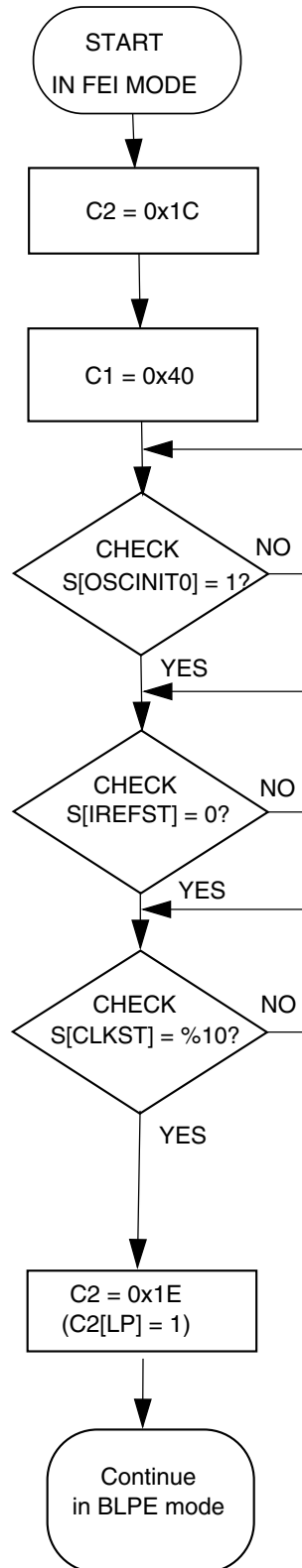
1. FLL\_R is the reference divider selected by the C1[FRDIV] bits, F is the FLL factor selected by C4[DRST\_DRS] and C4[DMX32] bits.

This section will include three mode switching examples using an MHz external crystal.

### 25.5.3.1 Example 1: Moving from FEI to BLPE mode: External Crystal = 4 MHz, MCGOUTCLK frequency = 4 MHz

In this example, the MCG will move through the proper operational modes from FEI to BLPE to achieve 4 MHz MCGOUTCLK frequency from 4 MHz external crystal reference. First, the code sequence will be described. Then there is a flowchart that illustrates the sequence.

1. First, FEI must transition to FBE mode:
  - a.  $C2 = 0x1C$ 
    - $C2[RANGE0]$  set to  $2'b01$  because the frequency of 4 MHz is within the high frequency range.
    - $C2[HGO0]$  set to 1 to configure the crystal oscillator for high gain operation.
    - $C2[EREFS0]$  set to 1, because a crystal is being used.
  - b.  $C1 = 0x90$ 
    - $C1[CLKS]$  set to  $2'b10$  to select external reference clock as system clock source
    - $C1[FRDIV]$  set to  $3'b010$ , or divide-by-128 because  $4\text{ MHz} / 128 = 31.25\text{ kHz}$  which is in the 31.25 kHz to 39.0625 kHz range required by the FLL
    - $C1[IREFS]$  cleared to 0, selecting the external reference clock and enabling the external oscillator.
  - c. Loop until  $S[OSCINIT0]$  is 1, indicating the crystal selected by  $C2[EREFS0]$  has been initialized.
  - d. Loop until  $S[IREFST]$  is 0, indicating the external reference is the current source for the reference clock.
  - e. Loop until  $S[CLKST]$  is  $2'b10$ , indicating that the external reference clock is selected to feed MCGOUTCLK.
2. Then, transition to BLPE:
  - a. Set  $C2[LP]$  to 1.



**Figure 25-14. Flowchart of FEI to BLPE mode transition using a 4 MHz crystal**  
 K02F Sub-Family Reference Manual , Rev. 1, 7/2014

### 25.5.3.2 Example 2: Moving from BLPE to BLPI mode: MCGOUTCLK frequency = 2 MHz

In this example, the MCG will move through the proper operational modes from BLPE mode with a 4 MHz crystal configured for a 4 MHz MCGOUTCLK frequency (see previous example) to BLPI mode with a 2 MHz MCGOUTCLK frequency. First, the code sequence will be described. Then there is a flowchart that illustrates the sequence.

1. First, BLPE must transition to FBE mode:
  - a. Clear C2[LP] to 0 here to switch to FBE mode.
2. Next, FBE mode transitions into FBI mode:
  - a. C1 = 0x54
    - C1[CLKS] set to 2'b01 to switch the system clock to the internal reference clock.
    - C1[IREFS] set to 1 to select the internal reference clock as the reference clock source.
    - C1[FRDIV] remain unchanged because the reference divider does not affect the internal reference.
  - b. Loop until S[IREFST] is 1, indicating the internal reference clock has been selected as the reference clock source.
  - c. Loop until S[CLKST] are 2'b01, indicating that the internal reference clock is selected to feed MCGOUTCLK.
3. Lastly, FBI transitions into BLPI mode with IRCS selecting Fast Internal Reference Clock.
  - a. C2 = 0x03
    - C2[IRCS] is 1
    - C2[LP] is 1
    - C2[RANGE], C2[HGO], C2[EREFS], C1[IRCLKEN], and C1[IREFSTEN] bits are ignored when the C1[IREFS] bit is set. They can remain set, or be cleared at this point.
  - b. Loop until S[IRCST] is 1, indicating the internal reference clock is the fast clock.

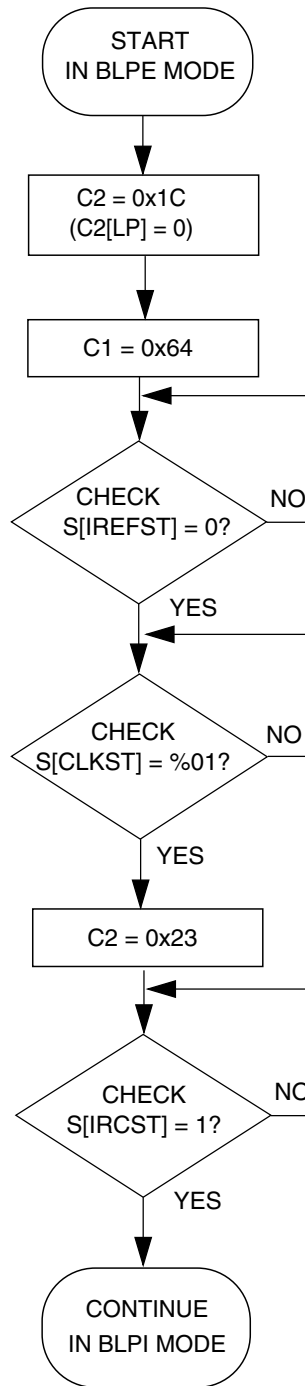


Figure 25-15. Flowchart of BLPE to BLPI mode transition using an MHz crystal

### 25.5.3.3 Example 3: Moving from BLPI to FEE mode

In this example, the MCG will move through the proper operational modes from BLPI mode at a 32 kHz MCGOUTCLK frequency running off the internal reference clock (see previous example) to FEE mode using a 4 MHz crystal configured for a 20 MHz MCGOUTCLK frequency. First, the code sequence will be described. Then there is a flowchart that illustrates the sequence.

1. First, BLPI must transition to FBI mode.
  - a.  $C2 = 0x00$ 
    - $C2[LP]$  is 0
2. Next, FBI will transition to FEE mode.
  - a.  $C2 = 0x1C$ 
    - $C2[RANGE]$  set to 2'b01 because the frequency of 4 MHz is within the high frequency range.
    - $C2[HGO]$  set to 1 to configure the crystal oscillator for high gain operation.
    - $C2[EREFS]$  set to 1, because a crystal is being used.
  - b.  $C1 = 0x10$ 
    - $C1[CLKS]$  set to 2'b00 to select the output of the FLL as system clock source.
    - $C1[FRDIV]$  remain at 3'b010, or divide-by-128 for a reference of 4 MHz / 128 = 31.25 kHz.
    - $C1[IREFS]$  cleared to 0, selecting the external reference clock.
  - c. Loop until  $S[OSCINIT]$  is 1, indicating the crystal selected by the  $C2[EREFS]$  bit has been initialized.
  - d. Loop until  $S[IREFST]$  is 0, indicating the external reference clock is the current source for the reference clock.
  - e. Loop until  $S[CLKST]$  are 2'b00, indicating that the output of the FLL is selected to feed MCGOUTCLK.
  - f. Now, with a 31.25 kHz reference frequency, a fixed DCO multiplier of 640,  $MCGOUTCLK = 31.25 \text{ kHz} * 640 / 1 = 20 \text{ MHz}$ .
  - g. At this point, by default, the  $C4[DRST\_DRS]$  bits are set to 2'b00 and  $C4[DMX32]$  is cleared to 0. If the MCGOUTCLK frequency of 40 MHz is desired instead, set the  $C4[DRST\_DRS]$  bits to 0x01 to switch the FLL

multiplication factor from 640 to 1280. To return the MCGOUTCLK frequency to 20 MHz, set C4[DRST\_DRS] bits to 2'b00 again, and the FLL multiplication factor will switch back to 640.

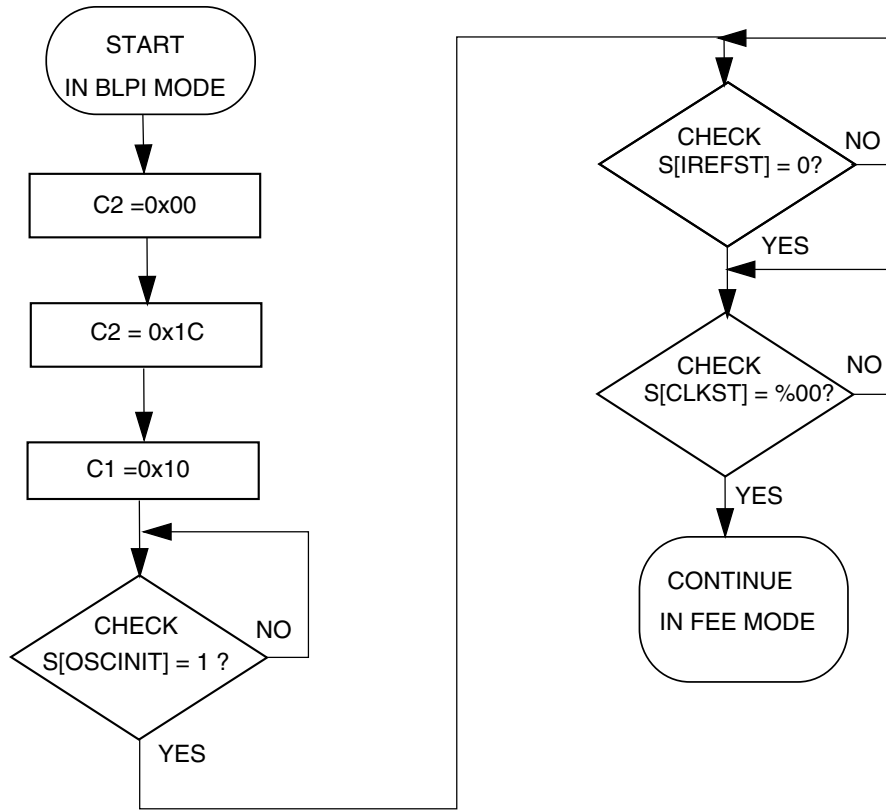


Figure 25-16. Flowchart of BLPI to FEE mode transition using an 4 MHz crystal



# Chapter 26

## Oscillator (OSC)

### 26.1 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.

### 26.2 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.

## 26.3 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.

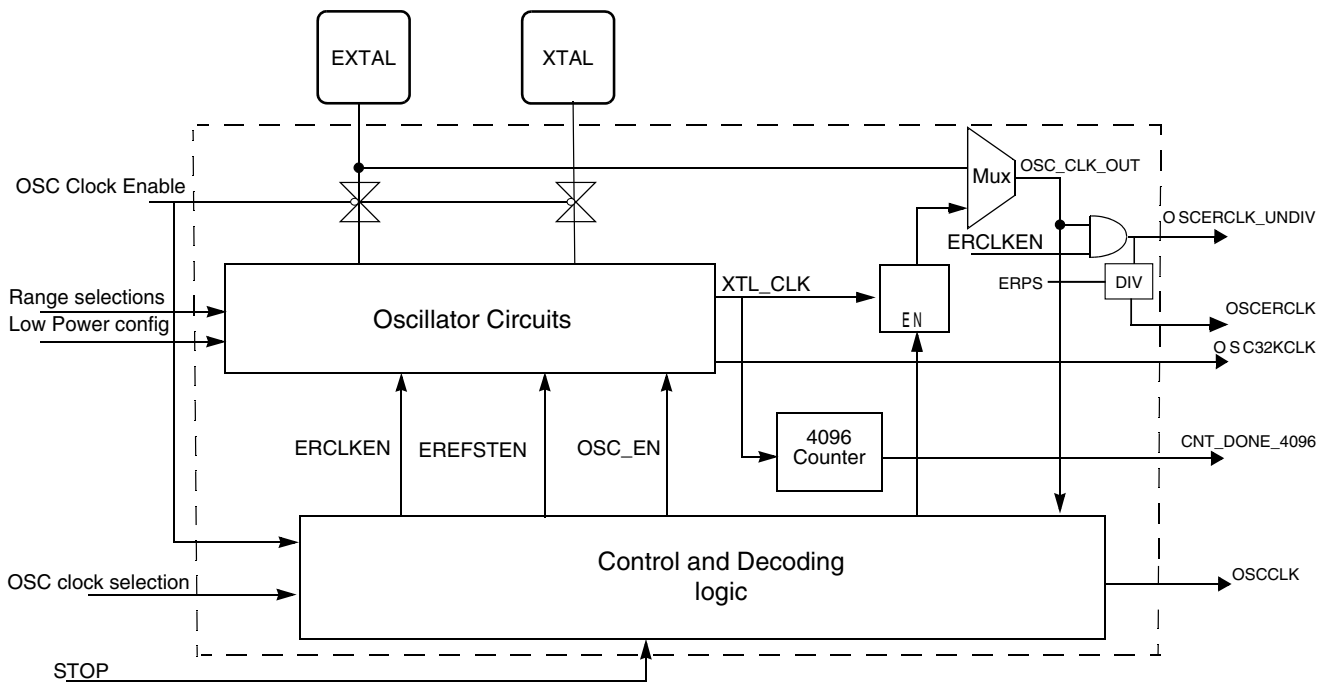


Figure 26-1. OSC Module Block Diagram

## 26.4 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 26-1. OSC Signal Descriptions**

Signal	Description	I/O
EXTAL	External clock/Oscillator input	I
XTAL	Oscillator output	O

## 26.5 External Crystal / Resonator Connections

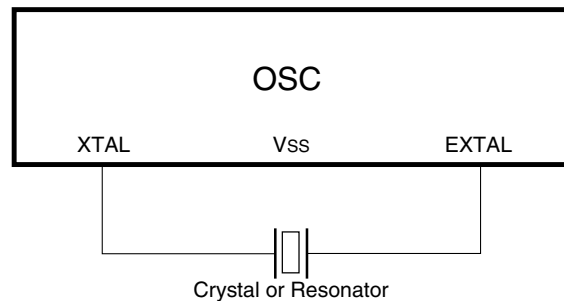
The connections for a crystal/resonator frequency reference are shown in the figures found here.

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 26-2. External Crystal/Resonator Connections**

Oscillator Mode	Connections
Low-frequency (32 kHz), low-power	Connection 1
Low-frequency (32 kHz), high-gain	Connection 2/Connection 3 <sup>1</sup>
High-frequency (3~32 MHz), low-power	Connection 1/Connection 3 <sup>2,2</sup>
High-frequency (3~32 MHz), high-gain	Connection 2/Connection 3 <sup>2</sup>

1. When the load capacitors ( $C_x$ ,  $C_y$ ) are greater than 30 pF, use Connection 3.
2. 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.



**Figure 26-2. Crystal/Ceramic Resonator Connections - Connection 1**

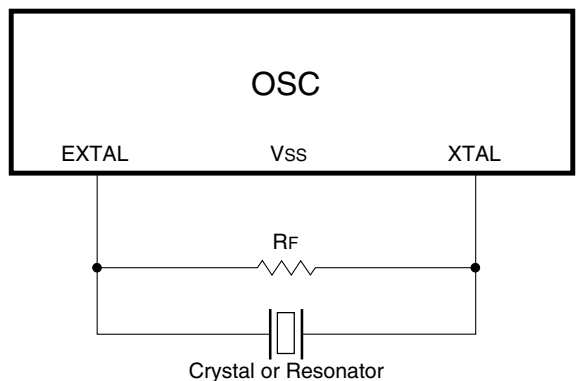


Figure 26-3. Crystal/Ceramic Resonator Connections - Connection 2

**NOTE**

Connection 1 and Connection 2 should use internal capacitors as the load of the oscillator by configuring the CR[SCxP] bits.

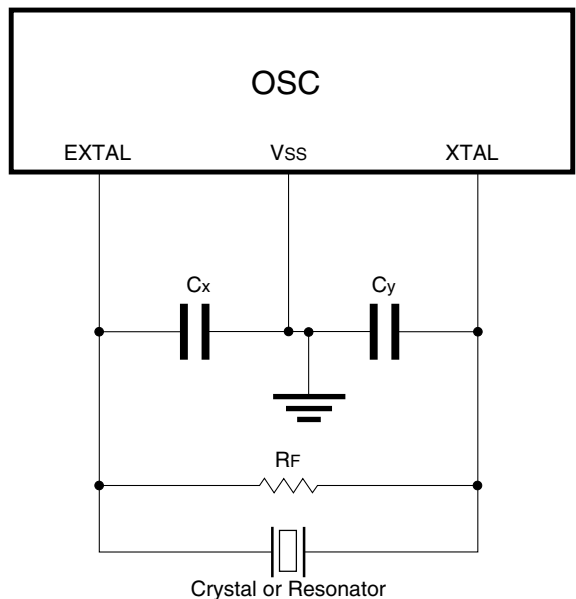


Figure 26-4. Crystal/Ceramic Resonator Connections - Connection 3

## 26.6 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.

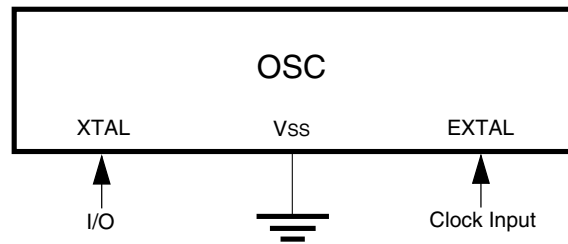


Figure 26-5. External Clock Connections

## 26.7 Memory Map/Register Definitions

Some oscillator module register bits are typically incorporated into other peripherals such as MCG or SIM.

### 26.7.1 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	<a href="#">26.71.1/501</a>
4006_5002	OSC_DIV (OSC_OSC_DIV)	8	R/W	00h	<a href="#">26.71.2/503</a>

#### 26.71.1 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.

## 26.71.2 OSC\_DIV (OSC\_OSC\_DIV)

OSC CLock divider register.

Address: 4006\_5000h base + 2h offset = 4006\_5002h

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

### OSC\_OSC\_DIV field descriptions

Field	Description
7-6 ERPS	ERCLK prescaler. These two bits are used to divide the ERCLK output. The un-divided ERCLK output is not affected by these two bits.  00 The divisor ratio is 1. 01 The divisor ratio is 2. 10 The divisor ratio is 4. 11 The divisor ratio is 8.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 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 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.

## 26.8 Functional Description

Functional details of the module can be found here.

### 26.8.1 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.

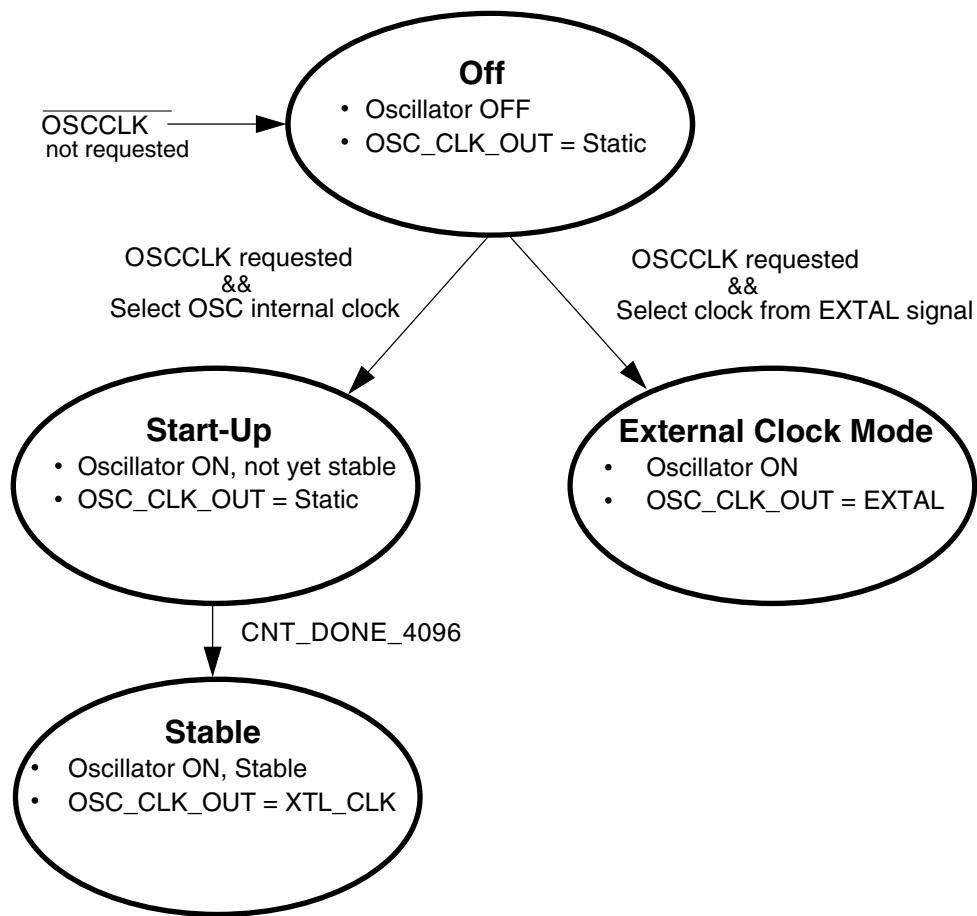


Figure 26-8. OSC Module state diagram

**NOTE**

XTL\_CLK is the clock generated internally from OSC circuits.

**26.8.1.1 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.



### 26.8.1.2 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.

### 26.8.1.3 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.

### 26.8.1.4 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.

## 26.8.2 OSC module modes

The OSC is a pierce-type oscillator that supports external crystals or resonators operating over the frequency ranges shown in [Table 26-6](#). 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 26-6. Oscillator modes**

Mode	Frequency Range
Low-frequency, high-gain	$f_{osc\_lo}$ (32.768 kHz) up to $f_{osc\_lo}$ (39.0625 kHz)

*Table continues on the next page...*

**Table 26-6. Oscillator modes (continued)**

Mode	Frequency Range
High-frequency mode1, high-gain	$f_{\text{osc\_hi\_1}}$ (3 MHz) up to $f_{\text{osc\_hi\_1}}$ (8 MHz)
High-frequency mode1, low-power	
High-frequency mode2, high-gain	$f_{\text{osc\_hi\_2}}$ (8 MHz) up to $f_{\text{osc\_hi\_2}}$ (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.

**26.8.2.1 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.

**26.8.2.2 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 feedback resistor that biases EXTAL.

### 26.8.2.3 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. In this mode, the internal capacitors could be used.

### 26.8.2.4 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, the internal capacitors could be used, the internal feedback resistor is connected, and 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.

## 26.8.3 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.

## 26.8.4 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.

## 26.9 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.

## 26.10 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.

In Low Leakage Stop (LLS) modes, the OSC holds all register settings. If CR[ERCLKEN] and CR[EREFSTEN] are set before entry to Low Leakage Stop modes, the OSC is still functional in these modes. After waking up from Very Low Leakage Stop (VLLSx) modes, all OSC register bits are reset and initialization is required through software.

## 26.11 Interrupts

The OSC module does not generate any interrupts.

# Chapter 27

## Flash Memory Controller (FMC)

### 27.1 Introduction

#### NOTE

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

The Flash Memory Controller (FMC) is a memory acceleration unit that provides:

- an interface between the device and the nonvolatile memory.
- buffers that can accelerate flash memory transfers.

#### 27.1.1 Overview

The Flash Memory Controller manages the interface between the device and the flash memory. The FMC receives status information detailing the configuration of the memory and uses this information to ensure a proper interface. The following table shows the supported read/write operations.

Flash memory type	Read	Write
Program flash memory	8-bit, 16-bit, and 32-bit reads	— <sup>1</sup>

1. A write operation to program flash memory results in a bus error.

In addition, for bank 0 and bank 1, the FMC provides three separate mechanisms for accelerating the interface between the device and the flash memory. A 64-bit speculation buffer can prefetch the next 64-bit flash memory location, and both a 4-way, 8-set cache and a single-entry 64-bit buffer can store previously accessed flash memory data for quick access times.

## 27.1.2 Features

The FMC's features include:

- Interface between the device and the flash memory:
  - 8-bit, 16-bit, and 32-bit read operations to program flash memory.
  - For bank 0 and bank 1: Read accesses to consecutive 32-bit spaces in memory return the second read data with no wait states. The memory returns 64 bits via the 32-bit bus access.
  - Crossbar master access protection for setting no access, read-only access, write-only access, or read/write access for each crossbar master.
- For bank 0 and bank 1: Acceleration of data transfer from program flash memory to the device:
  - 64-bit prefetch speculation buffer with controls for instruction/data access per master and bank
  - 4-way, 8-set, 64-bit line size cache for a total of thirty-two 64-bit entries with controls for replacement algorithm and lock per way for each bank
  - Single-entry buffer per bank
  - Invalidation control for the speculation buffer and the single-entry buffer

## 27.2 Modes of operation

The FMC only operates when a bus master accesses the flash memory.

For any device power mode where the flash memory cannot be accessed, the FMC is disabled.

## 27.3 External signal description

The FMC has no external signals.

## 27.4 Memory map and register descriptions

The programming model consists of the FMC control registers and the program visible cache (data and tag/valid entries).

### NOTE

Program the registers only while the flash controller is idle (for example, execute from RAM). Changing configuration settings

while a flash access is in progress can lead to non-deterministic behavior.

**Table 27-2. FMC register access**

Registers	Read access		Write access	
	Mode	Length	Mode	Length
Control registers: PFAPR, PFB0CR, PFB1CR	Supervisor (privileged) mode or user mode	32 bits	Supervisor (privileged) mode only	32 bits
Cache registers	Supervisor (privileged) mode or user mode	32 bits	Supervisor (privileged) mode only	32 bits

### NOTE

Accesses to unimplemented registers within the FMC's 4 KB address space return a bus error.

The cache entries, both data and tag/valid, can be read at any time.

### NOTE

System software is required to maintain memory coherence when any segment of the flash cache is programmed. For example, all buffer data associated with the reprogrammed flash should be invalidated. Accordingly, cache program visible writes must occur after a programming or erase event is completed and before the new memory image is accessed.

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. The following table elaborates on the tag/valid and data entries.

**Table 27-3. Program visible cache registers**

Cache storage	Based at offset	Contents of 32-bit read	Nomenclature	Nomenclature example
Tag	100h	13'h0, tag[18:5], 4'h0, valid	In TAGVDWxSy, x denotes the way and y denotes the set.	TAGVDW2S0 is the 14-bit tag and 1-bit valid for cache entry way 2, set 0.
Data	200h	Upper or lower longword of data	In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively.	DATAW1S0U represents bits [63:32] of data entry way 1, set 0, and DATAW1S0L represents bits [31:0] of data entry way 1, set 0.

## FMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4001_F000	Flash Access Protection Register (FMC_PFAPR)	32	R/W	00F8_003Fh	<a href="#">27.4.1/515</a>
4001_F004	Flash Bank 0 Control Register (FMC_PFB0CR)	32	R/W	3002_001Fh	<a href="#">27.4.2/519</a>
4001_F008	Flash Bank 1 Control Register (FMC_PFB1CR)	32	R/W	3002_001Fh	<a href="#">27.4.3/522</a>
4001_F100	Cache Tag Storage (FMC_TAGVDW0S0)	32	R/W	0000_0000h	<a href="#">27.4.4/524</a>
4001_F104	Cache Tag Storage (FMC_TAGVDW0S1)	32	R/W	0000_0000h	<a href="#">27.4.4/524</a>
4001_F108	Cache Tag Storage (FMC_TAGVDW0S2)	32	R/W	0000_0000h	<a href="#">27.4.4/524</a>
4001_F10C	Cache Tag Storage (FMC_TAGVDW0S3)	32	R/W	0000_0000h	<a href="#">27.4.4/524</a>
4001_F110	Cache Tag Storage (FMC_TAGVDW0S4)	32	R/W	0000_0000h	<a href="#">27.4.4/524</a>
4001_F114	Cache Tag Storage (FMC_TAGVDW0S5)	32	R/W	0000_0000h	<a href="#">27.4.4/524</a>
4001_F118	Cache Tag Storage (FMC_TAGVDW0S6)	32	R/W	0000_0000h	<a href="#">27.4.4/524</a>
4001_F11C	Cache Tag Storage (FMC_TAGVDW0S7)	32	R/W	0000_0000h	<a href="#">27.4.4/524</a>
4001_F120	Cache Tag Storage (FMC_TAGVDW1S0)	32	R/W	0000_0000h	<a href="#">27.4.5/525</a>
4001_F124	Cache Tag Storage (FMC_TAGVDW1S1)	32	R/W	0000_0000h	<a href="#">27.4.5/525</a>
4001_F128	Cache Tag Storage (FMC_TAGVDW1S2)	32	R/W	0000_0000h	<a href="#">27.4.5/525</a>
4001_F12C	Cache Tag Storage (FMC_TAGVDW1S3)	32	R/W	0000_0000h	<a href="#">27.4.5/525</a>
4001_F130	Cache Tag Storage (FMC_TAGVDW1S4)	32	R/W	0000_0000h	<a href="#">27.4.5/525</a>
4001_F134	Cache Tag Storage (FMC_TAGVDW1S5)	32	R/W	0000_0000h	<a href="#">27.4.5/525</a>
4001_F138	Cache Tag Storage (FMC_TAGVDW1S6)	32	R/W	0000_0000h	<a href="#">27.4.5/525</a>
4001_F13C	Cache Tag Storage (FMC_TAGVDW1S7)	32	R/W	0000_0000h	<a href="#">27.4.5/525</a>
4001_F140	Cache Tag Storage (FMC_TAGVDW2S0)	32	R/W	0000_0000h	<a href="#">27.4.6/526</a>
4001_F144	Cache Tag Storage (FMC_TAGVDW2S1)	32	R/W	0000_0000h	<a href="#">27.4.6/526</a>
4001_F148	Cache Tag Storage (FMC_TAGVDW2S2)	32	R/W	0000_0000h	<a href="#">27.4.6/526</a>
4001_F14C	Cache Tag Storage (FMC_TAGVDW2S3)	32	R/W	0000_0000h	<a href="#">27.4.6/526</a>
4001_F150	Cache Tag Storage (FMC_TAGVDW2S4)	32	R/W	0000_0000h	<a href="#">27.4.6/526</a>
4001_F154	Cache Tag Storage (FMC_TAGVDW2S5)	32	R/W	0000_0000h	<a href="#">27.4.6/526</a>
4001_F158	Cache Tag Storage (FMC_TAGVDW2S6)	32	R/W	0000_0000h	<a href="#">27.4.6/526</a>
4001_F15C	Cache Tag Storage (FMC_TAGVDW2S7)	32	R/W	0000_0000h	<a href="#">27.4.6/526</a>
4001_F160	Cache Tag Storage (FMC_TAGVDW3S0)	32	R/W	0000_0000h	<a href="#">27.4.7/527</a>
4001_F164	Cache Tag Storage (FMC_TAGVDW3S1)	32	R/W	0000_0000h	<a href="#">27.4.7/527</a>
4001_F168	Cache Tag Storage (FMC_TAGVDW3S2)	32	R/W	0000_0000h	<a href="#">27.4.7/527</a>
4001_F16C	Cache Tag Storage (FMC_TAGVDW3S3)	32	R/W	0000_0000h	<a href="#">27.4.7/527</a>
4001_F170	Cache Tag Storage (FMC_TAGVDW3S4)	32	R/W	0000_0000h	<a href="#">27.4.7/527</a>
4001_F174	Cache Tag Storage (FMC_TAGVDW3S5)	32	R/W	0000_0000h	<a href="#">27.4.7/527</a>
4001_F178	Cache Tag Storage (FMC_TAGVDW3S6)	32	R/W	0000_0000h	<a href="#">27.4.7/527</a>
4001_F17C	Cache Tag Storage (FMC_TAGVDW3S7)	32	R/W	0000_0000h	<a href="#">27.4.7/527</a>
4001_F200	Cache Data Storage (upper word) (FMC_DATAW0S0U)	32	R/W	0000_0000h	<a href="#">27.4.8/527</a>
4001_F204	Cache Data Storage (lower word) (FMC_DATAW0S0L)	32	R/W	0000_0000h	<a href="#">27.4.9/528</a>
4001_F208	Cache Data Storage (upper word) (FMC_DATAW0S1U)	32	R/W	0000_0000h	<a href="#">27.4.8/527</a>

Table continues on the next page...



## FMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4001_F20C	Cache Data Storage (lower word) (FMC_DATAW0S1L)	32	R/W	0000_0000h	<a href="#">27.4.9/528</a>
4001_F210	Cache Data Storage (upper word) (FMC_DATAW0S2U)	32	R/W	0000_0000h	<a href="#">27.4.8/527</a>
4001_F214	Cache Data Storage (lower word) (FMC_DATAW0S2L)	32	R/W	0000_0000h	<a href="#">27.4.9/528</a>
4001_F218	Cache Data Storage (upper word) (FMC_DATAW0S3U)	32	R/W	0000_0000h	<a href="#">27.4.8/527</a>
4001_F21C	Cache Data Storage (lower word) (FMC_DATAW0S3L)	32	R/W	0000_0000h	<a href="#">27.4.9/528</a>
4001_F220	Cache Data Storage (upper word) (FMC_DATAW0S4U)	32	R/W	0000_0000h	<a href="#">27.4.8/527</a>
4001_F224	Cache Data Storage (lower word) (FMC_DATAW0S4L)	32	R/W	0000_0000h	<a href="#">27.4.9/528</a>
4001_F228	Cache Data Storage (upper word) (FMC_DATAW0S5U)	32	R/W	0000_0000h	<a href="#">27.4.8/527</a>
4001_F22C	Cache Data Storage (lower word) (FMC_DATAW0S5L)	32	R/W	0000_0000h	<a href="#">27.4.9/528</a>
4001_F230	Cache Data Storage (upper word) (FMC_DATAW0S6U)	32	R/W	0000_0000h	<a href="#">27.4.8/527</a>
4001_F234	Cache Data Storage (lower word) (FMC_DATAW0S6L)	32	R/W	0000_0000h	<a href="#">27.4.9/528</a>
4001_F238	Cache Data Storage (upper word) (FMC_DATAW0S7U)	32	R/W	0000_0000h	<a href="#">27.4.8/527</a>
4001_F23C	Cache Data Storage (lower word) (FMC_DATAW0S7L)	32	R/W	0000_0000h	<a href="#">27.4.9/528</a>
4001_F240	Cache Data Storage (upper word) (FMC_DATAW1S0U)	32	R/W	0000_0000h	<a href="#">27.4.10/528</a>
4001_F244	Cache Data Storage (lower word) (FMC_DATAW1S0L)	32	R/W	0000_0000h	<a href="#">27.4.11/529</a>
4001_F248	Cache Data Storage (upper word) (FMC_DATAW1S1U)	32	R/W	0000_0000h	<a href="#">27.4.10/528</a>
4001_F24C	Cache Data Storage (lower word) (FMC_DATAW1S1L)	32	R/W	0000_0000h	<a href="#">27.4.11/529</a>
4001_F250	Cache Data Storage (upper word) (FMC_DATAW1S2U)	32	R/W	0000_0000h	<a href="#">27.4.10/528</a>
4001_F254	Cache Data Storage (lower word) (FMC_DATAW1S2L)	32	R/W	0000_0000h	<a href="#">27.4.11/529</a>
4001_F258	Cache Data Storage (upper word) (FMC_DATAW1S3U)	32	R/W	0000_0000h	<a href="#">27.4.10/528</a>
4001_F25C	Cache Data Storage (lower word) (FMC_DATAW1S3L)	32	R/W	0000_0000h	<a href="#">27.4.11/529</a>
4001_F260	Cache Data Storage (upper word) (FMC_DATAW1S4U)	32	R/W	0000_0000h	<a href="#">27.4.10/528</a>
4001_F264	Cache Data Storage (lower word) (FMC_DATAW1S4L)	32	R/W	0000_0000h	<a href="#">27.4.11/529</a>
4001_F268	Cache Data Storage (upper word) (FMC_DATAW1S5U)	32	R/W	0000_0000h	<a href="#">27.4.10/528</a>
4001_F26C	Cache Data Storage (lower word) (FMC_DATAW1S5L)	32	R/W	0000_0000h	<a href="#">27.4.11/529</a>
4001_F270	Cache Data Storage (upper word) (FMC_DATAW1S6U)	32	R/W	0000_0000h	<a href="#">27.4.10/528</a>
4001_F274	Cache Data Storage (lower word) (FMC_DATAW1S6L)	32	R/W	0000_0000h	<a href="#">27.4.11/529</a>

Table continues on the next page...

## FMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4001_F278	Cache Data Storage (upper word) (FMC_DATAW1S7U)	32	R/W	0000_0000h	<a href="#">27.4.10/528</a>
4001_F27C	Cache Data Storage (lower word) (FMC_DATAW1S7L)	32	R/W	0000_0000h	<a href="#">27.4.11/529</a>
4001_F280	Cache Data Storage (upper word) (FMC_DATAW2S0U)	32	R/W	0000_0000h	<a href="#">27.4.12/529</a>
4001_F284	Cache Data Storage (lower word) (FMC_DATAW2S0L)	32	R/W	0000_0000h	<a href="#">27.4.13/530</a>
4001_F288	Cache Data Storage (upper word) (FMC_DATAW2S1U)	32	R/W	0000_0000h	<a href="#">27.4.12/529</a>
4001_F28C	Cache Data Storage (lower word) (FMC_DATAW2S1L)	32	R/W	0000_0000h	<a href="#">27.4.13/530</a>
4001_F290	Cache Data Storage (upper word) (FMC_DATAW2S2U)	32	R/W	0000_0000h	<a href="#">27.4.12/529</a>
4001_F294	Cache Data Storage (lower word) (FMC_DATAW2S2L)	32	R/W	0000_0000h	<a href="#">27.4.13/530</a>
4001_F298	Cache Data Storage (upper word) (FMC_DATAW2S3U)	32	R/W	0000_0000h	<a href="#">27.4.12/529</a>
4001_F29C	Cache Data Storage (lower word) (FMC_DATAW2S3L)	32	R/W	0000_0000h	<a href="#">27.4.13/530</a>
4001_F2A0	Cache Data Storage (upper word) (FMC_DATAW2S4U)	32	R/W	0000_0000h	<a href="#">27.4.12/529</a>
4001_F2A4	Cache Data Storage (lower word) (FMC_DATAW2S4L)	32	R/W	0000_0000h	<a href="#">27.4.13/530</a>
4001_F2A8	Cache Data Storage (upper word) (FMC_DATAW2S5U)	32	R/W	0000_0000h	<a href="#">27.4.12/529</a>
4001_F2AC	Cache Data Storage (lower word) (FMC_DATAW2S5L)	32	R/W	0000_0000h	<a href="#">27.4.13/530</a>
4001_F2B0	Cache Data Storage (upper word) (FMC_DATAW2S6U)	32	R/W	0000_0000h	<a href="#">27.4.12/529</a>
4001_F2B4	Cache Data Storage (lower word) (FMC_DATAW2S6L)	32	R/W	0000_0000h	<a href="#">27.4.13/530</a>
4001_F2B8	Cache Data Storage (upper word) (FMC_DATAW2S7U)	32	R/W	0000_0000h	<a href="#">27.4.12/529</a>
4001_F2BC	Cache Data Storage (lower word) (FMC_DATAW2S7L)	32	R/W	0000_0000h	<a href="#">27.4.13/530</a>
4001_F2C0	Cache Data Storage (upper word) (FMC_DATAW3S0U)	32	R/W	0000_0000h	<a href="#">27.4.14/530</a>
4001_F2C4	Cache Data Storage (lower word) (FMC_DATAW3S0L)	32	R/W	0000_0000h	<a href="#">27.4.15/531</a>
4001_F2C8	Cache Data Storage (upper word) (FMC_DATAW3S1U)	32	R/W	0000_0000h	<a href="#">27.4.14/530</a>
4001_F2CC	Cache Data Storage (lower word) (FMC_DATAW3S1L)	32	R/W	0000_0000h	<a href="#">27.4.15/531</a>

Table continues on the next page...

## FMC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4001_F2D0	Cache Data Storage (upper word) (FMC_DATAW3S2U)	32	R/W	0000_0000h	27.4.14/ 530
4001_F2D4	Cache Data Storage (lower word) (FMC_DATAW3S2L)	32	R/W	0000_0000h	27.4.15/ 531
4001_F2D8	Cache Data Storage (upper word) (FMC_DATAW3S3U)	32	R/W	0000_0000h	27.4.14/ 530
4001_F2DC	Cache Data Storage (lower word) (FMC_DATAW3S3L)	32	R/W	0000_0000h	27.4.15/ 531
4001_F2E0	Cache Data Storage (upper word) (FMC_DATAW3S4U)	32	R/W	0000_0000h	27.4.14/ 530
4001_F2E4	Cache Data Storage (lower word) (FMC_DATAW3S4L)	32	R/W	0000_0000h	27.4.15/ 531
4001_F2E8	Cache Data Storage (upper word) (FMC_DATAW3S5U)	32	R/W	0000_0000h	27.4.14/ 530
4001_F2EC	Cache Data Storage (lower word) (FMC_DATAW3S5L)	32	R/W	0000_0000h	27.4.15/ 531
4001_F2F0	Cache Data Storage (upper word) (FMC_DATAW3S6U)	32	R/W	0000_0000h	27.4.14/ 530
4001_F2F4	Cache Data Storage (lower word) (FMC_DATAW3S6L)	32	R/W	0000_0000h	27.4.15/ 531
4001_F2F8	Cache Data Storage (upper word) (FMC_DATAW3S7U)	32	R/W	0000_0000h	27.4.14/ 530
4001_F2FC	Cache Data Storage (lower word) (FMC_DATAW3S7L)	32	R/W	0000_0000h	27.4.15/ 531

## 27.4.1 Flash Access Protection Register (FMC\_PFAPR)

Address: 4001\_F000h base + 0h offset = 4001\_F000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								M7PFD	M6PFD	M5PFD	M4PFD	M3PFD	M2PFD	M1PFD	M0PFD
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	M7AP[1:0]	M6AP[1:0]	M5AP[1:0]	M4AP[1:0]	M3AP[1:0]	M2AP[1:0]	M1AP[1:0]	M0AP[1:0]								
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

## FMC\_PFAPR field descriptions

Field	Description
31–24 Reserved	This field is reserved.
23 M7PFD	<p>Master 7 Prefetch Disable</p> <p>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.</p> <p>0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.</p>
22 M6PFD	<p>Master 6 Prefetch Disable</p> <p>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.</p> <p>0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.</p>
21 M5PFD	<p>Master 5 Prefetch Disable</p> <p>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.</p> <p>0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.</p>
20 M4PFD	<p>Master 4 Prefetch Disable</p> <p>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.</p> <p>0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.</p>
19 M3PFD	<p>Master 3 Prefetch Disable</p> <p>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.</p> <p>0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.</p>
18 M2PFD	<p>Master 2 Prefetch Disable</p> <p>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.</p> <p>0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.</p>
17 M1PFD	<p>Master 1 Prefetch Disable</p> <p>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.</p> <p>0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.</p>

*Table continues on the next page...*

## FMC\_PFAPR field descriptions (continued)

Field	Description
16 M0PFD	<p>Master 0 Prefetch Disable</p> <p>These bits control whether prefetching is enabled based on the logical number of the requesting crossbar switch master. This field is further qualified by the PFBnCR[BxDPE,BxIPE] bits.</p> <p>0 Prefetching for this master is enabled. 1 Prefetching for this master is disabled.</p>
15–14 M7AP[1:0]	<p>Master 7 Access Protection</p> <p>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.</p> <p>00 No access may be performed by this master. 01 Only read accesses may be performed by this master. 10 Only write accesses may be performed by this master. 11 Both read and write accesses may be performed by this master.</p>
13–12 M6AP[1:0]	<p>Master 6 Access Protection</p> <p>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.</p> <p>00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master</p>
11–10 M5AP[1:0]	<p>Master 5 Access Protection</p> <p>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.</p> <p>00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master</p>
9–8 M4AP[1:0]	<p>Master 4 Access Protection</p> <p>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.</p> <p>00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master</p>
7–6 M3AP[1:0]	<p>Master 3 Access Protection</p> <p>This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.</p> <p>00 No access may be performed by this master 01 Only read accesses may be performed by this master</p>

*Table continues on the next page...*

**FMC\_PFAPR field descriptions (continued)**

<b>Field</b>	<b>Description</b>
	10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
5–4 M2AP[1:0]	<b>Master 2 Access Protection</b>  This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.  00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
3–2 M1AP[1:0]	<b>Master 1 Access Protection</b>  This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.  00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
1–0 M0AP[1:0]	<b>Master 0 Access Protection</b>  This field controls whether read and write access to the flash are allowed based on the logical master number of the requesting crossbar switch master.  00 No access may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

## 27.4.2 Flash Bank 0 Control Register (FMC\_PFB0CR)

Address: 4001\_F000h base + 4h offset = 4001\_F004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	BORWSC[3:0]				CLCK_WAY[3:0]				0				0	B0MW[1:0]		0	
W									CINV_WAY[3:0]				S_B_INV				
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0								CRC[2:0]			B0DCE	B0ICE	B0DPE	B0IPE	B0SEBE	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	

### FMC\_PFB0CR field descriptions

Field	Description
31–28 BORWSC[3:0]	<p>Bank 0 Read Wait State Control</p> <p>This read-only field defines the number of wait states required to access the bank 0 flash memory.</p> <p>The relationship between the read access time of the flash array (expressed in system clock cycles) and RWSC is defined as:</p> <p>Access time of flash array [system clocks] = RWSC + 1</p> <p>The FMC automatically calculates this value based on the ratio of the system clock speed to the flash clock speed. For example, when this ratio is 4:1, the field's value is 3h.</p>
27–24 CLCK_WAY[3:0]	<p>Cache Lock Way x</p> <p>These bits determine if the given cache way is locked such that its contents will not be displaced by future misses.</p> <p>The bit setting definitions are for each bit in the field.</p> <p>0 Cache way is unlocked and may be displaced 1 Cache way is locked and its contents are not displaced</p>
23–20 CINV_WAY[3:0]	Cache Invalidate Way x

Table continues on the next page...

## FMC\_PFB0CR field descriptions (continued)

Field	Description
	<p>These bits determine if the given cache way is to be invalidated (cleared). When a bit within this field is written, the corresponding cache way is immediately invalidated: the way's tag, data, and valid contents are cleared. This field always reads as zero.</p> <p>Cache invalidation takes precedence over locking. The cache is invalidated by system reset. System software is required to maintain memory coherency when any segment of the flash memory is programmed or erased. Accordingly, cache invalidations must occur after a programming or erase event is completed and before the new memory image is accessed.</p> <p>The bit setting definitions are for each bit in the field.</p> <p>0 No cache way invalidation for the corresponding cache 1 Invalidate cache way for the corresponding cache: clear the tag, data, and vld bits of ways selected</p>
19 S_B_INV	<p>Invalidate Prefetch Speculation Buffer</p> <p>This bit determines if the FMC's prefetch speculation buffer and the single entry page buffer are to be invalidated (cleared). When this bit is written, the speculation buffer and single entry buffer are immediately cleared. This bit always reads as zero.</p> <p>0 Speculation buffer and single entry buffer are not affected. 1 Invalidate (clear) speculation buffer and single entry buffer.</p>
18–17 BOMW[1:0]	<p>Bank 0 Memory Width</p> <p>This read-only field defines the width of the bank 0 memory.</p> <p>00 32 bits 01 64 bits 10 Reserved 11 Reserved</p>
16 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
15–8 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
7–5 CRC[2:0]	<p>Cache Replacement Control</p> <p>This 3-bit field defines the replacement algorithm for accesses that are cached.</p> <p>000 LRU replacement algorithm per set across all four ways 001 Reserved 010 Independent LRU with ways [0-1] for ifetches, [2-3] for data 011 Independent LRU with ways [0-2] for ifetches, [3] for data 1xx Reserved</p>
4 B0DCE	<p>Bank 0 Data Cache Enable</p> <p>This bit controls whether data references are loaded into the cache.</p> <p>0 Do not cache data references. 1 Cache data references.</p>
3 B0ICE	<p>Bank 0 Instruction Cache Enable</p> <p>This bit controls whether instruction fetches are loaded into the cache.</p>

*Table continues on the next page...*



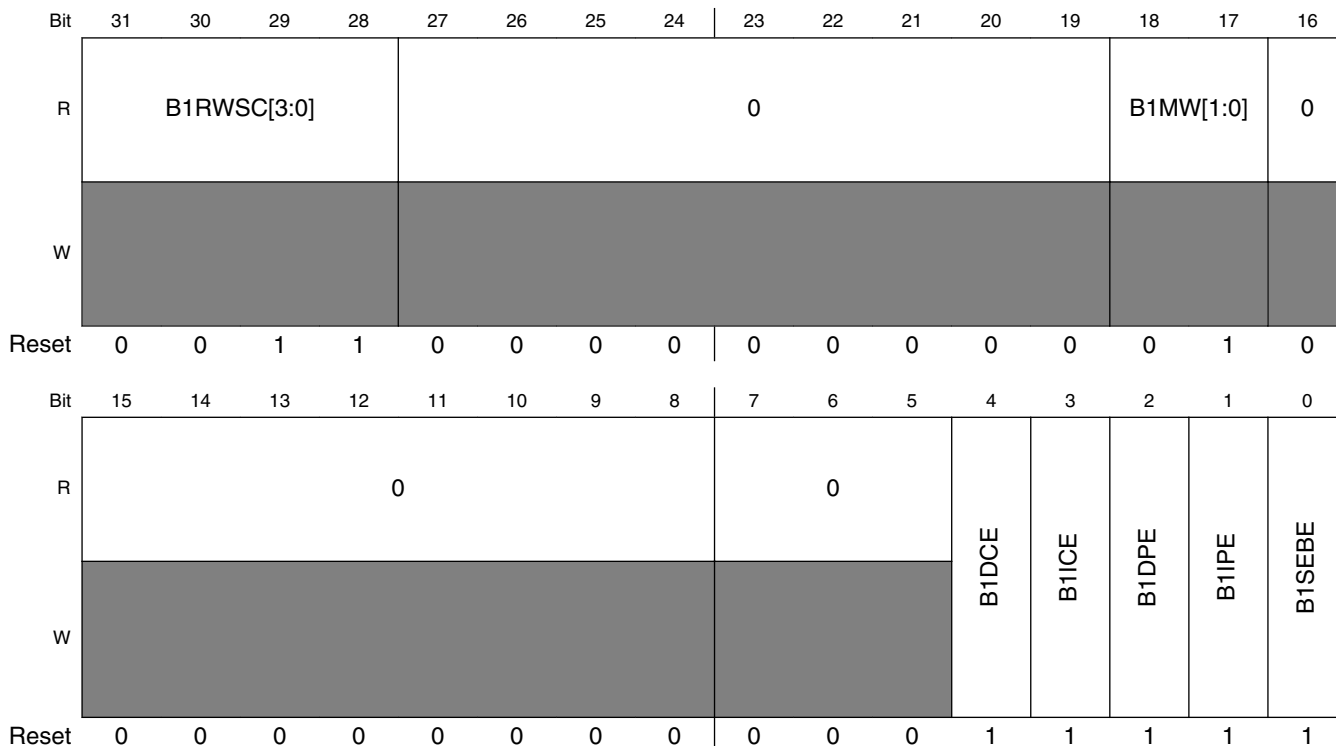
**FMC\_PFB0CR field descriptions (continued)**

<b>Field</b>	<b>Description</b>
	0 Do not cache instruction fetches. 1 Cache instruction fetches.
2 B0DPE	Bank 0 Data Prefetch Enable  This bit controls whether prefetches (or speculative accesses) are initiated in response to data references.  0 Do not prefetch in response to data references. 1 Enable prefetches in response to data references.
1 B0IPE	Bank 0 Instruction Prefetch Enable  This bit controls whether prefetches (or speculative accesses) are initiated in response to instruction fetches.  0 Do not prefetch in response to instruction fetches. 1 Enable prefetches in response to instruction fetches.
0 B0SEBE	Bank 0 Single Entry Buffer Enable  This bit controls whether the single entry page buffer is enabled in response to flash read accesses. Its operation is independent from bank 1's cache.  A high-to-low transition of this enable forces the page buffer to be invalidated.  0 Single entry buffer is disabled. 1 Single entry buffer is enabled.

### 27.4.3 Flash Bank 1 Control Register (FMC\_PFB1CR)

This register has a format similar to that for PFB0CR, except it controls the operation of flash bank 1, and the "global" cache control fields are empty.

Address: 4001\_F000h base + 8h offset = 4001\_F008h



#### FMC\_PFB1CR field descriptions

Field	Description
31–28 B1RWSC[3:0]	Bank 1 Read Wait State Control This read-only field defines the number of wait states required to access the bank 1 flash memory. The relationship between the read access time of the flash array (expressed in system clock cycles) and RWSC is defined as: Access time of flash array [system clocks] = RWSC + 1 The FMC automatically calculates this value based on the ratio of the system clock speed to the flash clock speed. For example, when this ratio is 4:1, the field's value is 3h.
27–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–17 B1MW[1:0]	Bank 1 Memory Width This read-only field defines the width of the bank 1 memory.

Table continues on the next page...

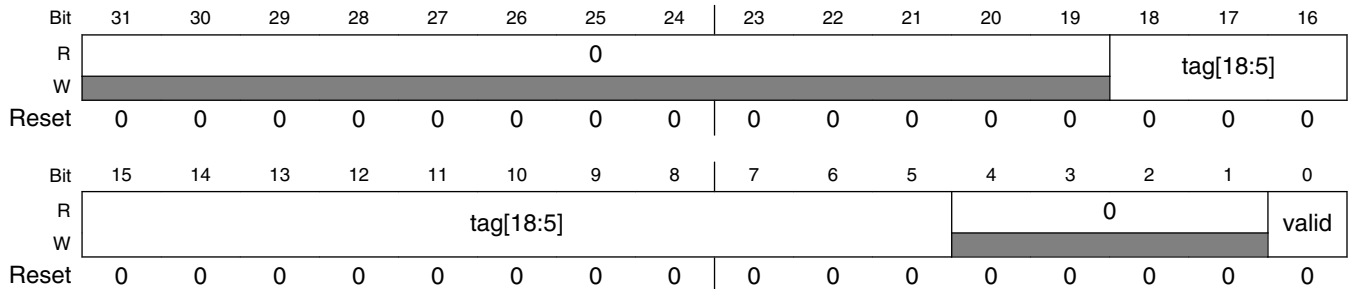
## FMC\_PFB1CR field descriptions (continued)

Field	Description
	00 32 bits 01 64 bits 10 Reserved 11 Reserved
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 B1DCE	Bank 1 Data Cache Enable  This bit controls whether data references are loaded into the cache.  0 Do not cache data references. 1 Cache data references.
3 B1ICE	Bank 1 Instruction Cache Enable  This bit controls whether instruction fetches are loaded into the cache.  0 Do not cache instruction fetches. 1 Cache instruction fetches.
2 B1DPE	Bank 1 Data Prefetch Enable  This bit controls whether prefetches (or speculative accesses) are initiated in response to data references.  0 Do not prefetch in response to data references. 1 Enable prefetches in response to data references.
1 B1IPE	Bank 1 Instruction Prefetch Enable  This bit controls whether prefetches (or speculative accesses) are initiated in response to instruction fetches.  0 Do not prefetch in response to instruction fetches. 1 Enable prefetches in response to instruction fetches.
0 B1SEBE	Bank 1 Single Entry Buffer Enable  This bit controls whether the single entry buffer is enabled in response to flash read accesses. Its operation is independent from bank 0's cache.  A high-to-low transition of this enable forces the page buffer to be invalidated.  0 Single entry buffer is disabled. 1 Single entry buffer is enabled.

### 27.4.4 Cache Tag Storage (FMC\_TAGVDW0Sn)

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In TAGVDWxSy, x denotes the way, and y denotes the set. This section represents tag/vld information for all sets in the indicated way.

Address: 4001\_F000h base + 100h offset + (4d × i), where i=0d to 7d



#### FMC\_TAGVDW0Sn field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–5 tag[18:5]	14-bit tag for cache entry
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 valid	1-bit valid for cache entry

## 27.4.5 Cache Tag Storage (FMC\_TAGVDW1Sn)

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In TAGVDW<sub>x</sub>S<sub>y</sub>, x denotes the way, and y denotes the set. This section represents tag/vld information for all sets in the indicated way.

Address: 4001\_F000h base + 120h offset + (4d × i), where i=0d to 7d

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

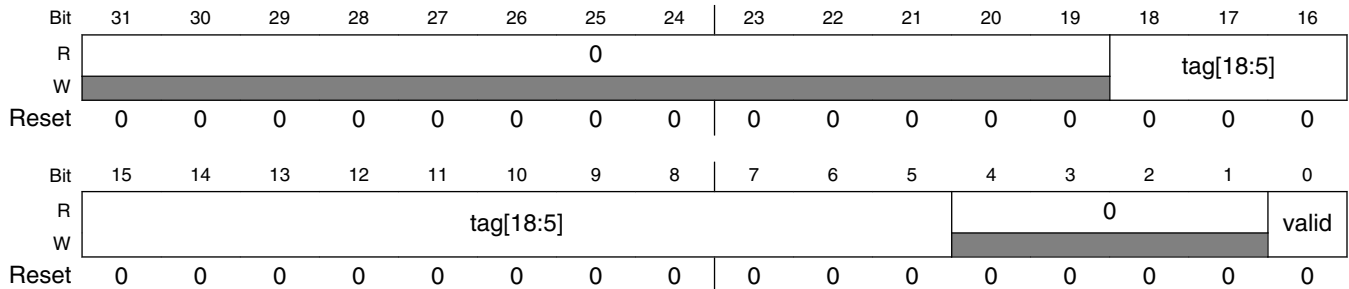
### FMC\_TAGVDW1Sn field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–5 tag[18:5]	14-bit tag for cache entry
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 valid	1-bit valid for cache entry

### 27.4.6 Cache Tag Storage (FMC\_TAGVDW2Sn)

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In TAGVDW<sub>x</sub>S<sub>y</sub>, x denotes the way, and y denotes the set. This section represents tag/vld information for all sets in the indicated way.

Address: 4001\_F000h base + 140h offset + (4d × i), where i=0d to 7d



#### FMC\_TAGVDW2Sn field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–5 tag[18:5]	14-bit tag for cache entry
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 valid	1-bit valid for cache entry

## 27.4.7 Cache Tag Storage (FMC\_TAGVDW3Sn)

The cache is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In TAGVDW<sub>x</sub>S<sub>y</sub>, x denotes the way, and y denotes the set. This section represents tag/vld information for all sets in the indicated way.

Address: 4001\_F000h base + 160h offset + (4d × i), where i=0d to 7d

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

**FMC\_TAGVDW3Sn field descriptions**

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–5 tag[18:5]	14-bit tag for cache entry
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 valid	1-bit valid for cache entry

## 27.4.8 Cache Data Storage (upper word) (FMC\_DATAW0SnU)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAW<sub>x</sub>S<sub>y</sub>U and DATAW<sub>x</sub>S<sub>y</sub>L, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the upper word (bits [63:32]) of all sets in the indicated way.

Address: 4001\_F000h base + 200h 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	data[63:32]																															
W	0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

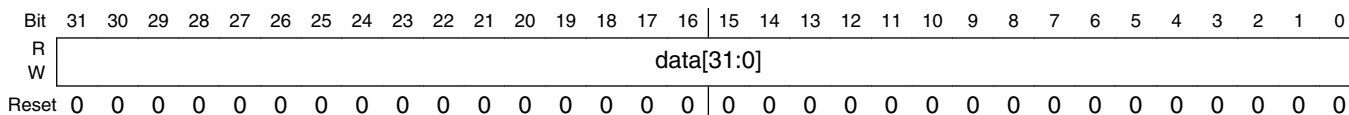
**FMC\_DATAW0SnU field descriptions**

Field	Description
31–0 data[63:32]	Bits [63:32] of data entry

**27.4.9 Cache Data Storage (lower word) (FMC\_DATAW0SnL)**

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the lower word (bits [31:0]) of all sets in the indicated way.

Address: 4001\_F000h base + 204h offset + (8d × i), where i=0d to 7d



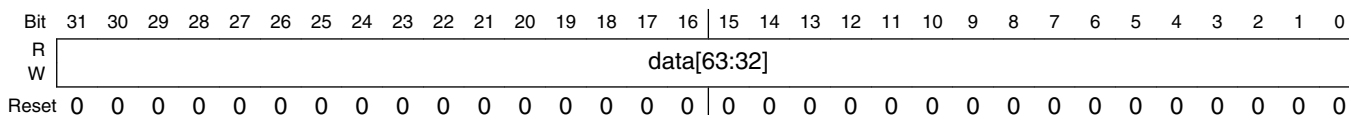
**FMC\_DATAW0SnL field descriptions**

Field	Description
31–0 data[31:0]	Bits [31:0] of data entry

**27.4.10 Cache Data Storage (upper word) (FMC\_DATAW1SnU)**

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the upper word (bits [63:32]) of all sets in the indicated way.

Address: 4001\_F000h base + 240h offset + (8d × i), where i=0d to 7d





### FMC\_DATAW1SnU field descriptions

Field	Description
31–0 data[63:32]	Bits [63:32] of data entry

#### 27.4.11 Cache Data Storage (lower word) (FMC\_DATAW1SnL)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the lower word (bits [31:0]) of all sets in the indicated way.

Address: 4001\_F000h base + 244h 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																																
W	data[31:0]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### FMC\_DATAW1SnL field descriptions

Field	Description
31–0 data[31:0]	Bits [31:0] of data entry

#### 27.4.12 Cache Data Storage (upper word) (FMC\_DATAW2SnU)

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the upper word (bits [63:32]) of all sets in the indicated way.

Address: 4001\_F000h base + 280h 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																																
W	data[63:32]																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

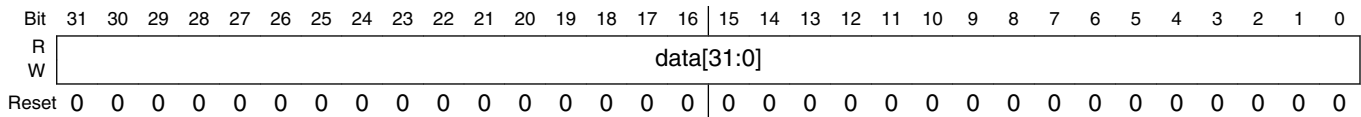
**FMC\_DATAW2SnU field descriptions**

Field	Description
31–0 data[63:32]	Bits [63:32] of data entry

**27.4.13 Cache Data Storage (lower word) (FMC\_DATAW2SnL)**

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the lower word (bits [31:0]) of all sets in the indicated way.

Address: 4001\_F000h base + 284h offset + (8d × i), where i=0d to 7d



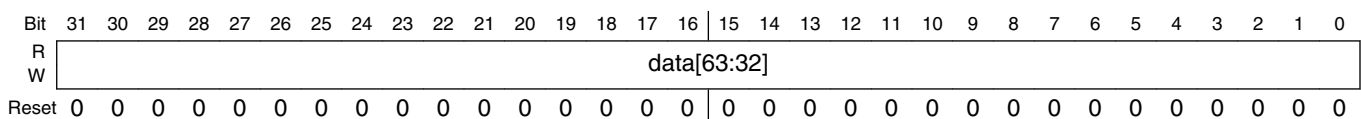
**FMC\_DATAW2SnL field descriptions**

Field	Description
31–0 data[31:0]	Bits [31:0] of data entry

**27.4.14 Cache Data Storage (upper word) (FMC\_DATAW3SnU)**

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAWxSyU and DATAWxSyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the upper word (bits [63:32]) of all sets in the indicated way.

Address: 4001\_F000h base + 2C0h offset + (8d × i), where i=0d to 7d



**FMC\_DATAW3SnU field descriptions**

Field	Description
31–0 data[63:32]	Bits [63:32] of data entry

**27.4.15 Cache Data Storage (lower word) (FMC\_DATAW3SnL)**

The cache of 64-bit entries is a 4-way, set-associative cache with 8 sets. The ways are numbered 0-3 and the sets are numbered 0-7. In DATAW<sub>x</sub>SyU and DATAW<sub>x</sub>SyL, x denotes the way, y denotes the set, and U and L represent upper and lower word, respectively. This section represents data for the lower word (bits [31:0]) of all sets in the indicated way.

Address: 4001\_F000h base + 2C4h 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																																
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

**FMC\_DATAW3SnL field descriptions**

Field	Description
31–0 data[31:0]	Bits [31:0] of data entry

**27.5 Functional description**

The FMC is a flash acceleration unit with flexible buffers for user configuration. Besides managing the interface between the device and the flash memory, the FMC can be used to restrict access from crossbar switch masters and—for program flash only—to customize the cache and buffers to provide single-cycle system-clock data-access times. Whenever a hit occurs for the prefetch speculation buffer, the cache, or the single-entry buffer, the requested data is transferred within a single system clock.

**27.5.1 Default configuration**

Upon system reset, the FMC is configured to provide a significant level of buffering for transfers from the flash memory:

- Crossbar masters 0, 1, 2 have read access to bank 0 and bank 1.
- For bank 0 and bank 1:
  - Prefetch support for data and instructions is enabled for crossbar masters 0, 1, 2.
  - The cache is configured for least recently used (LRU) replacement for all four ways.
  - The cache is configured for data or instruction replacement.
  - The single-entry buffer is enabled.

## 27.5.2 Configuration options

Though the default configuration provides a high degree of flash acceleration, advanced users may desire to customize the FMC buffer configurations to maximize throughput for their use cases. When reconfiguring the FMC for custom use cases, do not program the FMC's control registers while the flash memory is being accessed. Instead, change the control registers with a routine executing from RAM in supervisor mode.

The FMC's cache and buffering controls within PFB0CR and PFB1CR allow the tuning of resources to suit particular applications' needs. The cache and buffer are each controlled individually. The register controls enable buffering and prefetching per memory bank and access type (instruction fetch or data reference). The cache also supports 3 types of LRU replacement algorithms:

- LRU per set across all 4 ways,
- LRU with ways [0-1] for instruction fetches and ways [2-3] for data fetches, and
- LRU with ways [0-2] for instruction fetches and way [3] for data fetches.

As an application example: if both instruction fetches and data references are accessing flash memory, then control is available to send instruction fetches, data references, or both to the cache or the single-entry buffer. Likewise, speculation can be enabled or disabled for either type of access. If both instruction fetches and data references are cached, then the cache's way resources may be divided in several ways between the instruction fetches and data references.

In another application example, the cache can be configured for replacement from bank 0, while the single-entry buffer can be enabled for bank 1 only. This configuration is ideal for applications that use bank 0 for program space and bank 1 for data space.

### 27.5.3 Speculative reads

The FMC has a single buffer that reads ahead to the next word in the flash memory if there is an idle cycle. Speculative prefetching is programmable for each bank for instruction and/or data accesses using the B0DPE and B0IPE fields of PFB0CR and the B1DPE and B1IPE fields of PFB1CR. Because many code accesses are sequential, using the speculative prefetch buffer improves performance in most cases.

When speculative reads are enabled, the FMC immediately requests the next sequential address after a read completes. By requesting the next word immediately, speculative reads can help to reduce or even eliminate wait states when accessing sequential code and/or data.

For example, consider the following scenario:

- Assume a system with a 4:1 core-to-flash clock ratio and with speculative reads enabled.
- The core requests four sequential longwords in back-to-back requests, meaning there are no core cycle delays except for stalls waiting for flash memory data to be returned.
- None of the data is already stored in the cache or speculation buffer.

In this scenario, the sequence of events for accessing the four longwords is as follows:

1. The first longword read requires 4 to 7 core clocks. See [Wait states](#) for more information.
2. Due to the 64-bit data bus of the flash memory, the second longword read takes only 1 core clock because the data is already available inside the FMC. While the data for the second longword is being returned to the core, the FMC also starts reading the third and fourth longwords from the flash memory.
3. Accessing the third longword requires 3 core clock cycles. The flash memory read itself takes 4 clocks, but the first clock overlaps with the second longword read.
4. Reading the fourth longword, like the second longword, takes only 1 clock due to the 64-bit flash memory data bus.

## 27.6 Initialization and application information

The FMC does not require user initialization. Flash acceleration features are enabled by default.

## Initialization and application information

The FMC has no visibility into flash memory erase and program cycles because the Flash Memory module manages them directly. As a result, if an application is executing flash memory commands, the FMC's cache might need to be disabled and/or flushed to prevent the possibility of returning stale data. Use the PFB0CR[CINV\_WAY] field to invalidate the cache in this manner.

# Chapter 28

## Flash Memory Module (FTFA)

### 28.1 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.

## 28.1.1 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.

### 28.1.1.1 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

### 28.1.1.2 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

## 28.1.2 Block Diagram

The block diagram of the flash memory module is shown in the following figure.



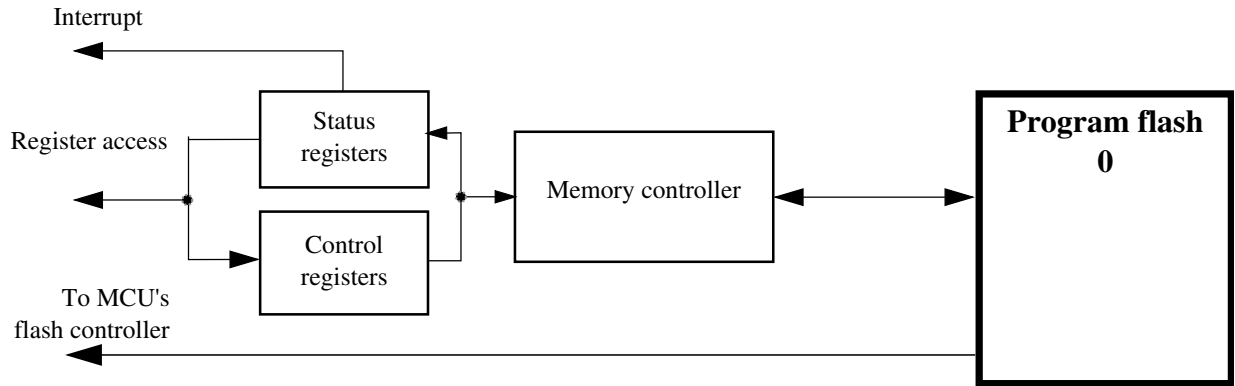


Figure 28-1. Flash Block Diagram

### 28.1.3 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.

**HSRUN** — An MCU power mode enabling high-speed access to the memory resources in the flash module. The user has no access to the flash command set when the MCU is in HSRUN mode.

**IFR** — Nonvolatile information register found in each flash block, separate from the main memory array.

**Longword** — 32 bits of data with an aligned longword having byte-address[1:0] = 00.

**NVM** — Nonvolatile memory. A memory technology that maintains stored data during power-off. The flash array is an NVM using NOR-type flash memory technology.

**NVM Normal Mode** — An NVM mode that provides basic user access to flash memory module resources. The CPU or other bus masters initiate flash program and erase operations (or other flash commands) using writes to the FCCOB register group in the flash memory module.

**NVM Special Mode** — An NVM mode enabling external, off-chip access to the memory resources in the flash memory module. A reduced flash command set is available when the MCU is secured. See the Chip Configuration details for information on when this mode is used.

**Phrase** — 64 bits of data with an aligned phrase having byte-address[2:0] = 000.

**Program flash** — The program flash memory provides nonvolatile storage for vectors and code store.

**Program flash Sector** — The smallest portion of the program flash memory (consecutive addresses) that can be erased.

**Retention** — The length of time that data can be kept in the NVM without experiencing errors upon readout. Since erased (1) states are subject to degradation just like programmed (0) states, the data retention limit may be reached from the last erase operation (not from the programming time).

**RWW**— Read-While-Write. The ability to simultaneously read from one memory resource while commanded operations are active in another memory resource.

**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.

## 28.2 External Signal Description

The flash memory module contains no signals that connect off-chip.

## 28.3 Memory Map and Registers

This section describes the memory map and registers for the flash 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.

### 28.3.1 Flash Configuration Field Description

The program flash memory contains a 16-byte flash configuration field that stores default protection settings (loaded on reset) and security information that allows the MCU to restrict access to the flash memory module.

Flash Configuration Field Byte Address	Size (Bytes)	Field Description
0x0_0400–0x0_0407	8	Backdoor Comparison Key. Refer to <a href="#">Verify Backdoor Access Key Command</a> and <a href="#">Unsecuring the Chip Using Backdoor Key Access</a> .
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).

### 28.3.2 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)

*Table continues on the next page...*

## Memory Map and Registers

Address Range	Size (Bytes)	Field Description
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)
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)

### 28.3.2.1 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](#)).

### 28.3.3 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	<a href="#">28.33.1/542</a>
4002_0001	Flash Configuration Register (FTFA_FCNFG)	8	R/W	00h	<a href="#">28.33.2/544</a>
4002_0002	Flash Security Register (FTFA_FSEC)	8	R	Undefined	<a href="#">28.33.3/545</a>
4002_0003	Flash Option Register (FTFA_FOPT)	8	R	Undefined	<a href="#">28.33.4/546</a>
4002_0004	Flash Common Command Object Registers (FTFA_FCCOB3)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_0005	Flash Common Command Object Registers (FTFA_FCCOB2)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_0006	Flash Common Command Object Registers (FTFA_FCCOB1)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_0007	Flash Common Command Object Registers (FTFA_FCCOB0)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_0008	Flash Common Command Object Registers (FTFA_FCCOB7)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_0009	Flash Common Command Object Registers (FTFA_FCCOB6)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_000A	Flash Common Command Object Registers (FTFA_FCCOB5)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_000B	Flash Common Command Object Registers (FTFA_FCCOB4)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_000C	Flash Common Command Object Registers (FTFA_FCCOBB)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_000D	Flash Common Command Object Registers (FTFA_FCCOBA)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_000E	Flash Common Command Object Registers (FTFA_FCCOB9)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_000F	Flash Common Command Object Registers (FTFA_FCCOB8)	8	R/W	00h	<a href="#">28.33.5/547</a>
4002_0010	Program Flash Protection Registers (FTFA_FPROT3)	8	R/W	Undefined	<a href="#">28.33.6/548</a>
4002_0011	Program Flash Protection Registers (FTFA_FPROT2)	8	R/W	Undefined	<a href="#">28.33.6/548</a>
4002_0012	Program Flash Protection Registers (FTFA_FPROT1)	8	R/W	Undefined	<a href="#">28.33.6/548</a>
4002_0013	Program Flash Protection Registers (FTFA_FPROT0)	8	R/W	Undefined	<a href="#">28.33.6/548</a>
4002_0018	Execute-only Access Registers (FTFA_XACCH3)	8	R	Undefined	<a href="#">28.33.7/550</a>
4002_0019	Execute-only Access Registers (FTFA_XACCH2)	8	R	Undefined	<a href="#">28.33.7/550</a>

Table continues on the next page...

## FTFA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4002_001A	Execute-only Access Registers (FTFA_XACCH1)	8	R	Undefined	28.33.7/ 550
4002_001B	Execute-only Access Registers (FTFA_XACCH0)	8	R	Undefined	28.33.7/ 550
4002_001C	Execute-only Access Registers (FTFA_XACCL3)	8	R	Undefined	28.33.7/ 550
4002_001D	Execute-only Access Registers (FTFA_XACCL2)	8	R	Undefined	28.33.7/ 550
4002_001E	Execute-only Access Registers (FTFA_XACCL1)	8	R	Undefined	28.33.7/ 550
4002_001F	Execute-only Access Registers (FTFA_XACCL0)	8	R	Undefined	28.33.7/ 550
4002_0020	Supervisor-only Access Registers (FTFA_SACCH3)	8	R	Undefined	28.33.8/ 551
4002_0021	Supervisor-only Access Registers (FTFA_SACCH2)	8	R	Undefined	28.33.8/ 551
4002_0022	Supervisor-only Access Registers (FTFA_SACCH1)	8	R	Undefined	28.33.8/ 551
4002_0023	Supervisor-only Access Registers (FTFA_SACCH0)	8	R	Undefined	28.33.8/ 551
4002_0024	Supervisor-only Access Registers (FTFA_SACCL3)	8	R	Undefined	28.33.8/ 551
4002_0025	Supervisor-only Access Registers (FTFA_SACCL2)	8	R	Undefined	28.33.8/ 551
4002_0026	Supervisor-only Access Registers (FTFA_SACCL1)	8	R	Undefined	28.33.8/ 551
4002_0027	Supervisor-only Access Registers (FTFA_SACCL0)	8	R	Undefined	28.33.8/ 551
4002_0028	Flash Access Segment Size Register (FTFA_FACSS)	8	R	Undefined	28.33.9/ 552
4002_002B	Flash Access Segment Number Register (FTFA_FACSN)	8	R	Undefined	28.33.10/ 553

### 28.33.1 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	<p>Command Complete Interrupt Flag</p> <p>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.</p> <p>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.</p> <p>0 Flash command in progress 1 Flash command has completed</p>
6 RDCOLERR	<p>Flash Read Collision Error Flag</p> <p>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.</p> <p>0 No collision error detected 1 Collision error detected</p>
5 ACCERR	<p>Flash Access Error Flag</p> <p>Indicates an illegal access has occurred to a flash memory resource caused by a violation of the command write sequence or issuing an illegal flash command. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. The ACCERR bit is cleared by writing a 1 to it. Writing a 0 to the ACCERR bit has no effect.</p> <p>0 No access error detected 1 Access error detected</p>
4 FPVIOL	<p>Flash Protection Violation Flag</p> <p>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 it. Writing a 0 to the FPVIOL bit has no effect.</p> <p>0 No protection violation detected 1 Protection violation detected</p>
3-1 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
0 MGSTAT0	<p>Memory Controller Command Completion Status Flag</p>

Table continues on the next page...

**FTFA\_FSTAT field descriptions (continued)**

Field	Description
	<p>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.</p> <p>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.</p>

**28.33.2 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	<p>Command Complete Interrupt Enable</p> <p>Controls interrupt generation when a flash command completes.</p> <p>0 Command complete interrupt disabled 1 Command complete interrupt enabled. An interrupt request is generated whenever the FSTAT[CCIF] flag is set.</p>
6 RDCOLLIE	<p>Read Collision Error Interrupt Enable</p> <p>Controls interrupt generation when a flash memory read collision error occurs.</p> <p>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]).</p>
5 ERSAREQ	<p>Erase All Request</p> <p>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.</p> <p>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.</p>

*Table continues on the next page...*



## FTFA\_FCENFG field descriptions (continued)

Field	Description
	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.

## 28.33.3 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.

Table continues on the next page...

## FTFA\_FSEC field descriptions (continued)

Field	Description
	00 Backdoor key access disabled 01 Backdoor key access disabled (preferred KEYEN state to disable backdoor key access) 10 Backdoor key access enabled 11 Backdoor key access disabled
5–4 MEEN	Mass Erase Enable Bits  Enables and disables mass erase capability of the flash memory module. The state of this field is relevant only when SEC is set to secure outside of NVM Normal Mode. When SEC is set to unsecure, the MEEN setting does not matter.  00 Mass erase is enabled 01 Mass erase is enabled 10 Mass erase is disabled 11 Mass erase is enabled
3–2 FSLACC	Freescale Failure Analysis Access Code  Enables or disables access to the flash memory contents during returned part failure analysis at Freescale. When SEC is secure and FSLACC is denied, access to the program flash contents is denied and any failure analysis performed by Freescale 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), Freescale 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 Freescale factory access granted 01 Freescale factory access denied 10 Freescale factory access denied 11 Freescale factory access granted
1–0 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.

### 28.33.4 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
7–0 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.

## 28.33.5 Flash Common Command Object Registers (FTFA\_FCCOBn)

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	CCOBn							
Write	CCOBn							
Reset	0	0	0	0	0	0	0	0

### FTFA\_FCCOBn field descriptions

Field	Description
7–0 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 queuing is provided; the next command can be loaded only after the current command completes.

**FTFA\_FCCOB $n$  field descriptions (continued)**

Field	Description																										
	<p>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.</p> <p>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.</p> <p><b>NOTE:</b> 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.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">FCCOB Number</th> <th style="text-align: center;">Typical Command Parameter Contents [7:0]</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>FCMD (a code that defines the flash command)</td> </tr> <tr> <td style="text-align: center;">1</td> <td>Flash address [23:16]</td> </tr> <tr> <td style="text-align: center;">2</td> <td>Flash address [15:8]</td> </tr> <tr> <td style="text-align: center;">3</td> <td>Flash address [7:0]</td> </tr> <tr> <td style="text-align: center;">4</td> <td>Data Byte 0</td> </tr> <tr> <td style="text-align: center;">5</td> <td>Data Byte 1</td> </tr> <tr> <td style="text-align: center;">6</td> <td>Data Byte 2</td> </tr> <tr> <td style="text-align: center;">7</td> <td>Data Byte 3</td> </tr> <tr> <td style="text-align: center;">8</td> <td>Data Byte 4</td> </tr> <tr> <td style="text-align: center;">9</td> <td>Data Byte 5</td> </tr> <tr> <td style="text-align: center;">A</td> <td>Data Byte 6</td> </tr> <tr> <td style="text-align: center;">B</td> <td>Data Byte 7</td> </tr> </tbody> </table> <p><b>FCCOB Endianness :</b></p> <p>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.</p>	FCCOB Number	Typical Command Parameter Contents [7:0]	0	FCMD (a code that defines the flash command)	1	Flash address [23:16]	2	Flash address [15:8]	3	Flash address [7:0]	4	Data Byte 0	5	Data Byte 1	6	Data Byte 2	7	Data Byte 3	8	Data Byte 4	9	Data Byte 5	A	Data Byte 6	B	Data Byte 7
FCCOB 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																										

**28.33.6 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 32 KB of program flash where each assigned bit protects 1 KB. For configurations with 24 KB of program flash memory or less, FPROT0 is not used. For configurations with 16

KB of program flash memory or less, FPROT1 is not used. For configurations with 8 KB of program flash memory, FPROT2 is not used. The bitfields are defined in each register as follows:

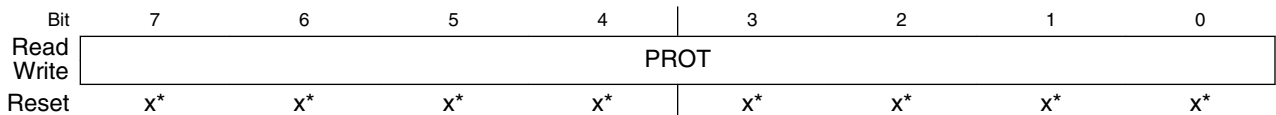
Program flash protection register	Program flash protection bits
FPROT0	PROT[31:24]
FPROT1	PROT[23:16]
FPROT2	PROT[15:8]
FPROT3	PROT[7:0]

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



\* Notes:

- x = Undefined at reset.

### FTFA\_FPROTn field descriptions

Field	Description
7-0 PROT	<p>Program Flash Region Protect</p> <p>Each program flash region can be protected from program and erase operations by setting the associated PROT bit.</p> <p><b>In NVM Normal mode:</b> 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.</p>

FTFA\_FPROT<sub>n</sub> field descriptions (continued)

Field	Description
	<p><b>In NVM Special mode:</b> All bits of FPROT are writable without restriction. Unprotected areas can be protected and protected areas can be unprotected.</p> <p><b>Restriction:</b> 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.</p> <p>Each bit in the 32-bit protection register represents 1/32 of the total program flash except for configurations where program flash memory is less than 32 KB. For configurations with less than 32 KB of program flash memory, each assigned bit represents 1 KB.</p> <p>0 Program flash region is protected. 1 Program flash region is not protected</p>

28.33.7 Execute-only Access Registers (FTFA\_XACC<sub>n</sub>)

The XACC registers define which program flash segments are restricted to data read or execute only or both data and instruction fetches.

The eight XACC registers allow up to 64 restricted segments of equal memory size.

Execute-only access register	Program flash execute-only access bits
XACCH0	XA[63:56]
XACCH1	XA[55:48]
XACCH2	XA[47:40]
XACCH3	XA[39:32]
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.

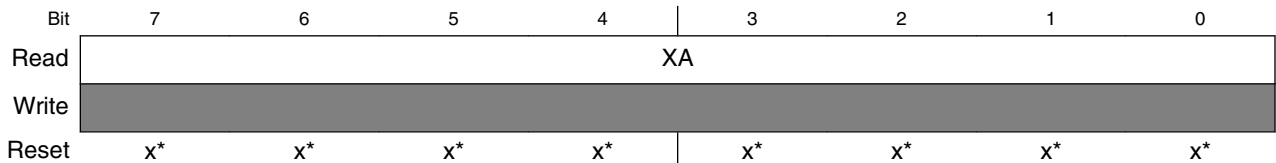
Execute-only access register	Program Flash IFR address A	Program Flash IFR address B
XACCH0	0xA3	0xAB
XACCH1	0xA2	0xAA
XACCH2	0xA1	0xA9
XACCH3	0xA0	0xA8
XACCL0	0xA7	0xAF
XACCL1	0xA6	0xAE

Table continues on the next page...

Execute-only access register	Program Flash IFR address A	Program Flash IFR address B
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 + 18h offset + (1d × i), where i=0d to 7d



\* Notes:

- x = Undefined at reset.

### FTFA\_XACC<sub>n</sub> field descriptions

Field	Description
7–0 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

## 28.33.8 Supervisor-only Access Registers (FTFA\_SACC<sub>n</sub>)

The SACC registers define which program flash segments are restricted to supervisor only or user and supervisor access.

The eight SACC registers allow up to 64 restricted segments of equal memory size.

Supervisor-only access register	Program flash supervisor-only access bits
SACCH0	SA[63:56]
SACCH1	SA[55:48]
SACCH2	SA[47:40]
SACCH3	SA[39:32]
SACCL0	SA[31:24]
SACCL1	SA[23:16]
SACCL2	SA[15:8]
SACCL3	SA[7:0]

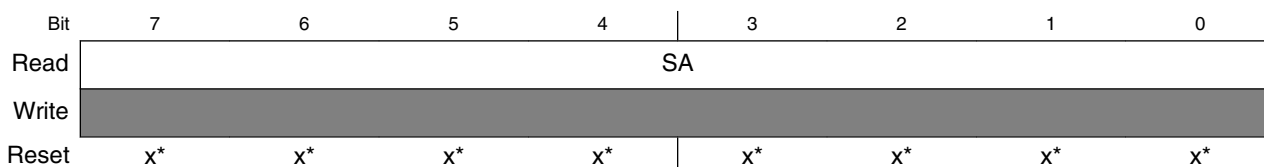
During the reset sequence, the SACC registers are loaded with the logical AND of Program Flash IFR addresses A and B as indicated in the following table.

## Memory Map and Registers

Supervisor-only access register	Program Flash IFR address A	Program Flash IFR address B
SACCH0	0xB3	0xBB
SACCH1	0xB2	0xBA
SACCH2	0xB1	0xB9
SACCH3	0xB0	0xB8
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 + 20h offset + (1d × i), where i=0d to 7d



\* Notes:

- x = Undefined at reset.

### FTFA\_SACCN field descriptions

Field	Description
7-0 SA	Supervisor-only access control
0	Associated segment is accessible in supervisor mode only
1	Associated segment is accessible in user or supervisor mode

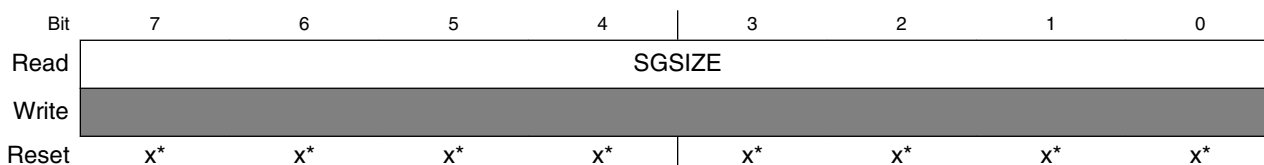
## 28.33.9 Flash Access Segment Size Register (FTFA\_FACSS)

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





\* Notes:

- x = Undefined at reset.

### FTFA\_FACSS field descriptions

Field	Description																		
7-0 SGSIZE	Segment Size The segment size is a fixed value based on the available program flash size divided by NUMSG.																		
	<table border="1"> <thead> <tr> <th>Program Flash Size</th> <th>Segment Size</th> <th>Segment Size Encoding</th> </tr> </thead> <tbody> <tr> <td>64 KBytes</td> <td>2 KBytes</td> <td>0x3</td> </tr> <tr> <td>128 KBytes</td> <td>4 KBytes</td> <td>0x4</td> </tr> <tr> <td>160 KBytes</td> <td>4 KBytes</td> <td>0x4</td> </tr> <tr> <td>256 KBytes</td> <td>4 KBytes</td> <td>0x4</td> </tr> <tr> <td>512 KBytes</td> <td>8 KBytes</td> <td>0x5</td> </tr> </tbody> </table>	Program Flash Size	Segment Size	Segment Size Encoding	64 KBytes	2 KBytes	0x3	128 KBytes	4 KBytes	0x4	160 KBytes	4 KBytes	0x4	256 KBytes	4 KBytes	0x4	512 KBytes	8 KBytes	0x5
Program Flash Size	Segment Size	Segment Size Encoding																	
64 KBytes	2 KBytes	0x3																	
128 KBytes	4 KBytes	0x4																	
160 KBytes	4 KBytes	0x4																	
256 KBytes	4 KBytes	0x4																	
512 KBytes	8 KBytes	0x5																	

### 28.33.10 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
7-0 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)

**FTFA\_FACSN field descriptions (continued)**

Field	Description
0x28	Program flash memory is divided into 40 segments (160 Kbytes)
0x40	Program flash memory is divided into 64 segments (256 Kbytes, 512 Kbytes)

## 28.4 Functional Description

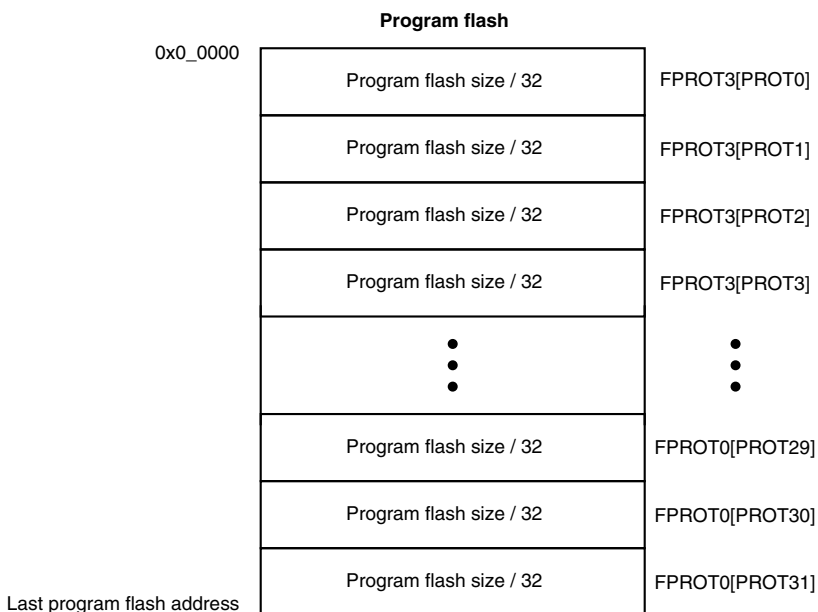
The information found here describes functional details of the flash memory module.

### 28.4.1 Flash Protection

Individual regions within the flash memory can be protected from program and erase operations.

Protection is controlled by the following registers:

- FPROT<sub>n</sub> —
  - For 2<sup>n</sup> program flash sizes, four registers typically protect 32 regions of the program flash memory as shown in the following figure



**Figure 28-44. Program flash protection**

**NOTE**

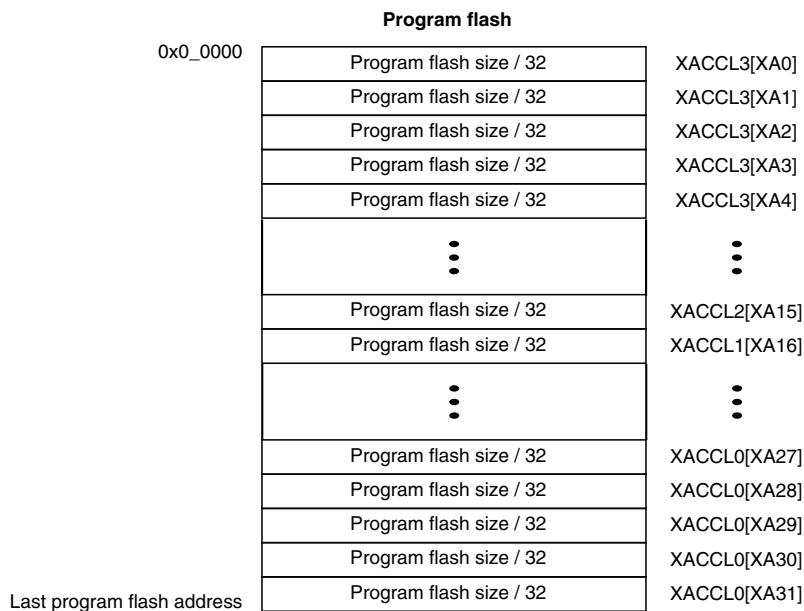
Flash protection features are discussed further in [AN4507: Using the Kinetis Security and Flash Protection Features](#) . Not

all features described in the application note are available on this device.

## 28.4.2 Flash Access Protection

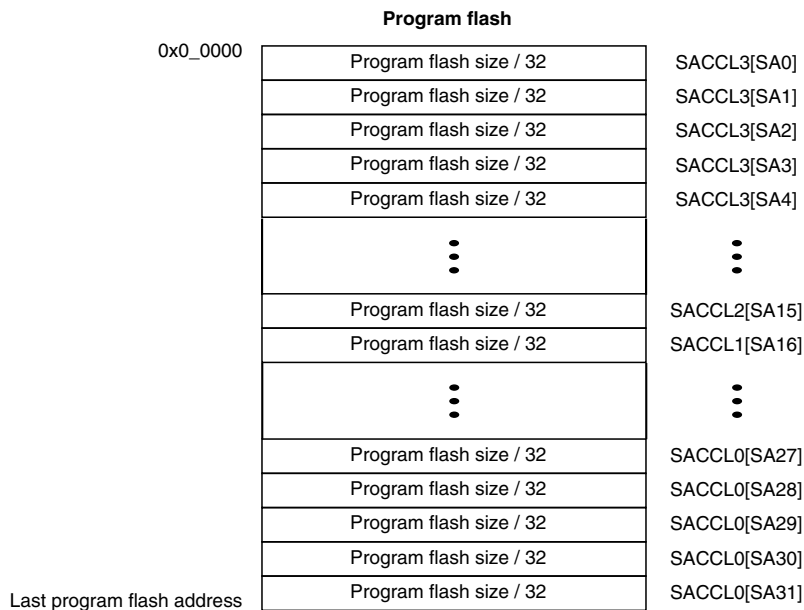
Individual segments within the program flash memory can be designated for restricted access. Specific flash commands monitor FXACC contents to protect flash memory but the FSACC contents do not impact flash command operation. Access is controlled by the following registers:

- FXACC —
  - 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 28-45. Program flash access control (64KB or 128KB of program flash)**

- FSACC —
  - 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 28-46. Program flash access control (64KB or 128KB of program flash)**

### 28.4.3 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 28-44. 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.

## 28.4.4 Flash Operation in Low-Power Modes

### 28.4.4.1 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](#)).

### 28.4.4.2 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.

## 28.4.5 Functional Modes of Operation

The flash memory module has two operating modes: NVM Normal and NVM Special.

The operating mode affects the command set availability (see [Table 28-45](#)). Refer to the Chip Configuration details of this device for how to activate each mode.

## 28.4.6 Flash Reads and Ignored Writes

The flash memory module requires only the flash address to execute a flash memory read.

The MCU must not read from the flash memory while commands are running (as evidenced by CCIF=0) on that block. Read data cannot be guaranteed from a flash block while any command is processing within that block. The block arbitration logic detects any simultaneous access and reports this as a read collision error (see the FSTAT[RDCOLERR] bit).

### 28.4.7 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).

### 28.4.8 Flash Program and Erase

All flash functions except read require the user to setup and launch a flash command through a series of peripheral bus writes.

The user cannot initiate any further flash commands until notified that the current command has completed. The flash command structure and operation are detailed in [Flash Command Operations](#).

### 28.4.9 Flash Command Operations

Flash command operations are typically used to modify flash memory contents.

The next sections describe:

- The command write sequence used to set flash command parameters and launch execution
- A description of all flash commands available

#### 28.4.9.1 Command Write Sequence

Flash commands are specified using a command write sequence illustrated in [Figure 28-47](#). 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. Attempts to launch a flash command in HSRUN mode will be trapped with the ACCERR flag being set.

### 28.4.9.1.1 Load the FCCOB Registers

The user must load the FCCOB registers with all parameters required by the desired flash command. The individual registers that make up the FCCOB data set can be written in any order.

### 28.4.9.1.2 Launch the Command by Clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by clearing FSTAT[CCIF] by writing a '1' to it. FSTAT[CCIF] remains 0 until the flash command completes.

The FSTAT register contains a blocking mechanism that prevents a new command from launching (can't clear FSTAT[CCIF]) if the previous command resulted in an access error (FSTAT[ACCERR]=1) or a protection violation (FSTAT[FPVIOL]=1). In error scenarios, two writes to FSTAT are required to initiate the next command: the first write clears the error flags, the second write clears CCIF.

### 28.4.9.1.3 Command Execution and Error Reporting

The command processing has several steps:

1. The flash memory module reads the command code and performs a series of parameter checks and protection checks, if applicable, which are unique to each command.

If the parameter check fails, the FSTAT[ACCERR] (access error) flag is set. FSTAT[ACCERR] reports invalid instruction codes and out-of bounds addresses. Usually, access errors suggest that the command was not set-up with valid parameters in the FCCOB register group.

Program and erase commands also check the address to determine if the operation is requested to execute on protected areas. If the protection check fails, FSTAT[FPVIOL] (protection error) flag is set.

Command processing never proceeds to execution when the parameter or protection step fails. Instead, command processing is terminated after setting FSTAT[CCIF].

2. If the parameter and protection checks pass, the command proceeds to execution. Run-time errors, such as failure to erase verify, may occur during the execution phase. Run-time errors are reported in FSTAT[MGSTAT0]. A command may have access errors, protection errors, and run-time errors, but the run-time errors are not seen until all access and protection errors have been corrected.
3. Command execution results, if applicable, are reported back to the user via the FCCOB and FSTAT registers.
4. The flash memory module sets FSTAT[CCIF] signifying that the command has completed.

The flow for a generic command write sequence is illustrated in the following figure.

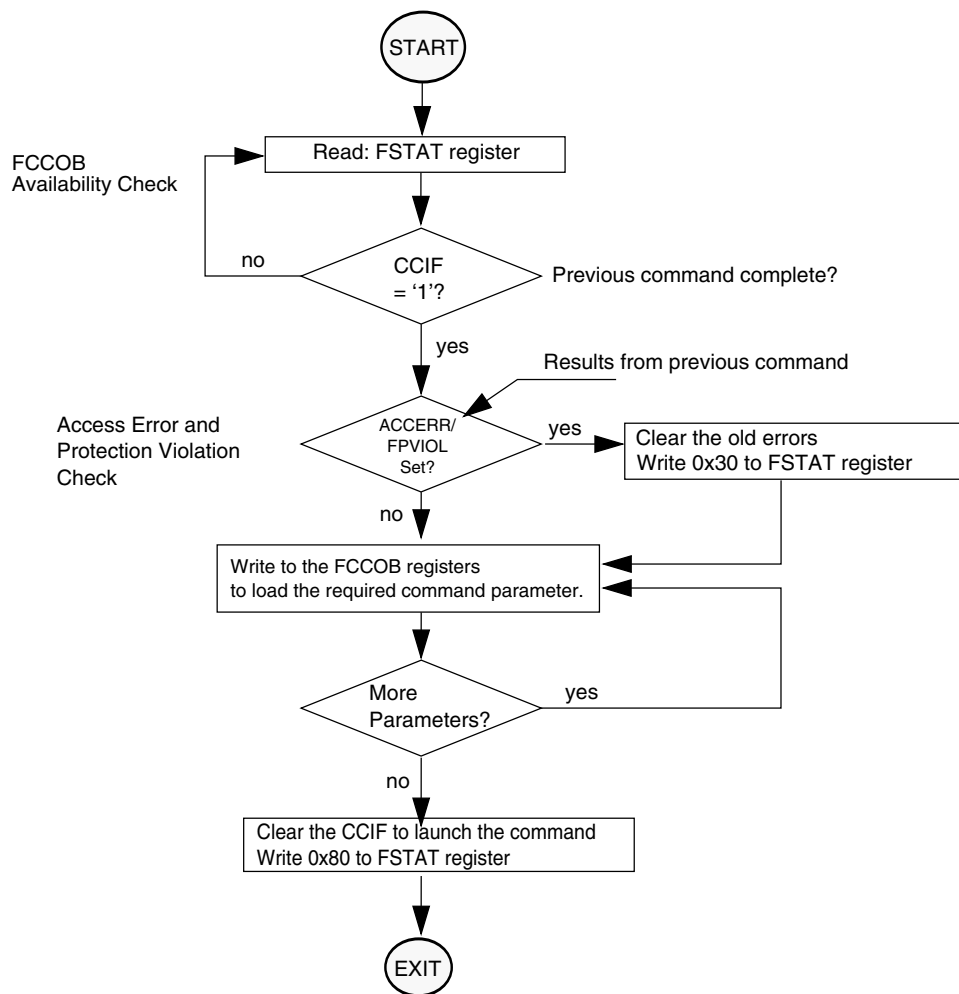


Figure 28-47. Generic flash command write sequence flowchart



### 28.4.9.2 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.  <b>NOTE:</b> 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.

### 28.4.9.3 Flash Commands by Mode

The following table shows the flash commands that can be executed in each flash operating mode.

**Table 28-45. Flash Commands by Mode**

FCMD	Command	NVM Normal			NVM Special		
		Unsecure	Secure	MEEN=10	Unsecure	Secure	MEEN=10
0x01	Read 1s Section	x	x	x	x	—	—
0x02	Program Check	x	x	x	x	—	—
0x03	Read Resource	x	x	x	x	—	—
0x06	Program Longword	x	x	x	x	—	—
0x09	Erase Flash Sector	x	x	x	x	—	—
0x40	Read 1s All Blocks	x	x	x	x	x	—
0x41	Read Once	x	x	x	x	—	—
0x43	Program Once	x	x	x	x	—	—
0x44	Erase All Blocks	x	x	x	x	x	—
0x45	Verify Backdoor Access Key	x	x	x	x	—	—

## 28.4.10 Margin Read Commands

The Read-1s commands (Read 1s All Blocks and Read 1s Section) and the Program Check command have a margin choice parameter that allows the user to apply non-standard read reference levels to the program flash array reads performed by these commands. Using the preset 'user' and 'factory' margin levels, these commands perform their associated read operations at tighter tolerances than a 'normal' read. These non-standard read levels are applied only during the command execution. All simple (uncommanded) flash array reads to the MCU always use the standard, un-margined, read reference level.

Only the 'normal' read level should be employed during normal flash usage. The non-standard, 'user' and 'factory' margin levels should be employed only in special cases. They can be used during special diagnostic routines to gain confidence that the device is not suffering from the end-of-life data loss customary of flash memory devices.

Erased ('1') and programmed ('0') bit states can degrade due to elapsed time and data cycling (number of times a bit is erased and re-programmed). The lifetime of the erased states is relative to the last erase operation. The lifetime of the programmed states is measured from the last program time.

The 'user' and 'factory' levels become, in effect, a minimum safety margin; i.e. if the reads pass at the tighter tolerances of the 'user' and 'factory' margins, then the 'normal' reads have at least this much safety margin before they experience data loss.

The 'user' margin is a small delta to the normal read reference level. 'User' margin levels can be employed to check that flash memory contents have adequate margin for normal level read operations. If unexpected read results are encountered when checking flash memory contents at the 'user' margin levels, loss of information might soon occur during 'normal' readout.

The 'factory' margin is a bigger deviation from the norm, a more stringent read criteria that should only be attempted immediately (or very soon) after completion of an erase or program command, early in the cycling life. 'Factory' margin levels can be used to check that flash memory contents have adequate margin for long-term data retention at the normal level setting. If unexpected results are encountered when checking flash memory contents at 'factory' margin levels, the flash memory contents should be erased and reprogrammed.

### CAUTION

Factory margin levels must only be used during verify of the initial factory programming.

## 28.4.11 Flash Command Description

This section describes all flash commands that can be launched by a command write sequence.

The flash memory module sets the FSTAT[ACCERR] bit and aborts the command execution if any of the following illegal conditions occur:

- There is an unrecognized command code in the FCCOB FCMD field.
- There is an error in a FCCOB field for the specific commands. Refer to the error handling table provided for each command.

Ensure that FSTAT[ACCERR] and FSTAT[FPVIOL] are cleared prior to starting the command write sequence. As described in [Launch the Command by Clearing CCIF](#), a new command cannot be launched while these error flags are set.

Do not attempt to read a flash block while the flash memory module is running a command (FSTAT[CCIF] = 0) on that same block. The flash memory module may return invalid data to the MCU with the collision error flag (FSTAT[RDCOLERR]) set.

### CAUTION

Flash data must be in the erased state before being programmed. Cumulative programming of bits (adding more zeros) is not allowed.

### 28.4.11.1 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 28-46. 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] <sup>1</sup> 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 28-47](#) 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 28-47. 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 28-48. 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]

### 28.4.11.2 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 28-49. 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] <sup>1</sup>
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 28-50](#), 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 28-50. 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 28-51. 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]

### 28.4.11.3 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 28-53](#).

**Table 28-52. 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] <sup>1</sup>
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 <a href="#">Table 28-53</a> )

1. Must be longword aligned (Flash address [1:0] = 00).

**Table 28-53. 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 <sup>1</sup>	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 28-54. 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]

#### 28.4.11.4 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 28-55. 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] <sup>1</sup>
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).

## Functional Description

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 28-56. 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]
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]

### 28.4.11.5 Erase Flash Sector Command

The Erase Flash Sector operation erases all addresses in a flash sector.

**Table 28-57. 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] <sup>1</sup> 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 28-48](#)).

**Table 28-58. 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 <sup>1</sup>	FSTAT[MGSTAT0]

1. User margin read may be run using the Read 1s Section command to verify all bits are erased.

### 28.4.11.5.1 Suspending an Erase Flash Sector Operation

To suspend an Erase Flash Sector operation set the FCNFG[ERSSUSP] bit (see [Flash Configuration Field Description](#)) when CCIF is clear and the CCOB command field holds the code for the Erase Flash Sector command. During the Erase Flash Sector operation (see [Erase Flash Sector Command](#)), the flash memory module samples the state of the ERSSUSP bit at convenient points. If the flash memory module detects that the ERSSUSP bit is set, the Erase Flash Sector operation is suspended and the flash memory module sets CCIF. While ERSSUSP is set, all writes to flash registers are ignored except for writes to the FSTAT and FCNFG registers.

If an Erase Flash Sector operation effectively completes before the flash memory module detects that a suspend request has been made, the flash memory module clears the ERSSUSP bit prior to setting CCIF. When an Erase Flash Sector operation has been successfully suspended, the flash memory module sets CCIF and leaves the ERSSUSP bit set. While CCIF is set, the ERSSUSP bit can only be cleared to prevent the withdrawal of a suspend request before the flash memory module has acknowledged it.

### 28.4.11.5.2 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 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.

### 28.4.11.5.3 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.

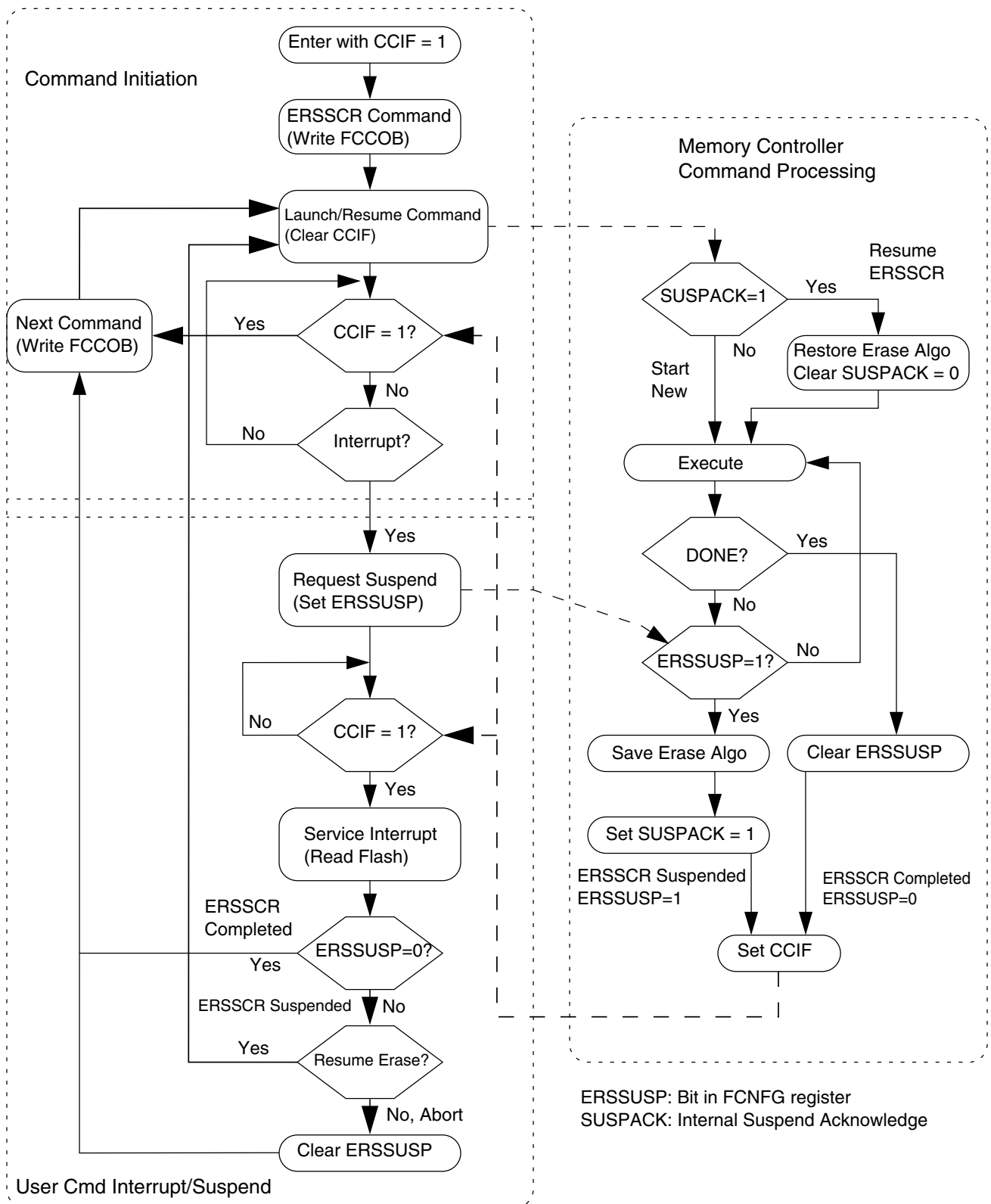


Figure 28-48. Suspend and Resume of Erase Flash Sector Operation

### 28.4.11.6 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 28-59. 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 28-60](#),
- 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 28-60. 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 28-61. Read 1s All Blocks Command Error Handling**

Error Condition	Error Bit
An invalid margin choice is specified	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

### 28.4.11.7 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 28-62. 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 28-63. 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]

### 28.4.11.8 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 28-64. 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 28-65. 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 <sup>1</sup>	FSTAT[ACCERR]

*Table continues on the next page...*

**Table 28-65. 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.

### 28.4.11.9 Erase All Blocks Command

The Erase All Blocks operation erases all flash memory, verifies all memory contents, and releases MCU security.

**Table 28-66. 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 28-67. 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 <sup>1</sup>	FSTAT[MGSTAT0]

1. User margin read may be run using the Read 1s All Blocks command to verify all bits are erased.

### 28.4.11.9.1 Triggering an Erase All External to the Flash Memory Module

The functionality of the Erase All BlocksErase All Blocks Unsecure command is also available in an uncommanded fashion outside of the flash memory. Refer to the device's Chip Configuration details for information on this functionality.

Before invoking the external erase all function, the FSTAT[ACCERR and PVIOL] flags must be cleared and the FCCOB0 register must not contain 0x44. When invoked, the erase-all function erases all program flash memory 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 as described in [Erase All Blocks Command](#).

### 28.4.11.10 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 28-68. 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 28-68. 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 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 28-69. Verify Backdoor Access Key Command Error Handling**

Error Condition	Error Bit
The supplied key is all-0s or all-Fs	FSTAT[ACCERR]
An incorrect backdoor key is supplied	FSTAT[ACCERR]
Backdoor key access has not been enabled (see the description of the FSEC register)	FSTAT[ACCERR]
This command is launched and the backdoor key has mismatched since the last power down reset	FSTAT[ACCERR]

## 28.4.12 Security

The flash memory module provides security information to the MCU based on contents of the FSEC security register.

The MCU then limits access to flash memory resources as defined in the device's Chip Configuration details. During reset, the flash memory module initializes the FSEC register using data read from the security byte of the Flash Configuration Field (see [Flash Configuration Field Description](#)).

The following fields are available in the FSEC register. The settings are described in the [Flash Security Register \(FTFA\\_FSEC\)](#) 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 28-70. FSEC register fields**

FSEC field	Description
KEYEN	Backdoor Key Access
MEEN	Mass Erase Capability
FSLACC	Freescale Factory Access
SEC	MCU security

### 28.4.12.1 Flash Memory Access by Mode and Security

The following table summarizes how access to the flash memory module is affected by security and operating mode.

**Table 28-71. Flash Memory Access Summary**

Operating Mode	Chip Security State	
	Unsecure	Secure
NVM Normal	Full command set	
NVM Special	Full command set	Only the Erase All Blocks and Read 1s All Blocks commands.

### 28.4.12.2 Changing the Security State

The security state out of reset can be permanently changed by programming the security byte of the flash configuration field. This assumes that you are starting from a mode where the necessary program flash erase and program commands are available and that the region of the program flash containing the flash configuration field is unprotected. If the flash security byte is successfully programmed, its new value takes affect after the next chip reset.

#### 28.4.12.2.1 Unsecuring the Chip Using Backdoor Key Access

The chip can be unsecured by using the backdoor key access feature, which requires knowledge of the contents of the 8-byte backdoor key value stored in the Flash Configuration Field (see [Flash Configuration Field Description](#)). If the FSEC[KEYEN] bits are in the enabled state, the Verify Backdoor Access Key command (see [Verify](#)

[Backdoor Access Key Command](#)) can be run; it allows the user to present prospective keys for comparison to the stored keys. If the keys match, the FSEC[SEC] bits are changed to unsecure the chip. The entire 8-byte key cannot be all 0s or all 1s; that is, 0000\_0000\_0000\_0000h and FFFF\_FFFF\_FFFF\_FFFFh are not accepted by the Verify Backdoor Access Key command as valid comparison values. While the Verify Backdoor Access Key command is active, program flash memory is not available for read access and returns invalid data.

The user code stored in the program flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN bits are in the enabled state, the chip can be unsecured by the following backdoor key access sequence:

1. Follow the command sequence for the Verify Backdoor Access Key command as explained in [Verify Backdoor Access Key Command](#)
2. If the Verify Backdoor Access Key command is successful, the chip is unsecured and the FSEC[SEC] bits are forced to the unsecure state

An illegal key provided to the Verify Backdoor Access Key command prohibits further use of the Verify Backdoor Access Key command. A reset of the chip is the only method to re-enable the Verify Backdoor Access Key command when a comparison fails.

After the backdoor keys have been correctly matched, the chip is unsecured by changing the FSEC[SEC] bits. A successful execution of the Verify Backdoor Access Key command changes the security in the FSEC register only. It does not alter the security byte or the keys stored in the Flash Configuration Field ([Flash Configuration Field Description](#)). After the next reset of the chip, the security state of the flash memory module reverts back to the flash security byte in the Flash Configuration Field. The Verify Backdoor Access Key command sequence has no effect on the program and erase protections defined in the program flash protection registers.

If the backdoor keys successfully match, the unsecured chip has full control of the contents of the Flash Configuration Field. The chip may erase the sector containing the Flash Configuration Field and reprogram the flash security byte to the unsecure state and change the backdoor keys to any desired value.

### 28.4.13 Reset Sequence

On each system reset the flash memory module executes a sequence which establishes initial values for the flash block configuration parameters, FPROT, FOPT, and FSEC registers.

## Functional Description

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 29

## Cyclic Redundancy Check (CRC)

### 29.1 Introduction

#### NOTE

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

The cyclic redundancy check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial, WAS, and other parameters required to implement a 16-bit or 32-bit CRC standard.

The 16/32-bit code is calculated for 32 bits of data at a time.

#### 29.1.1 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 byte-wise. This option is required for certain CRC standards. A byte-wise 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 byte-wise transpose function.
- Option for inversion of final CRC result
- 32-bit CPU register programming interface

#### 29.1.2 Block diagram

The following is a block diagram of the CRC.

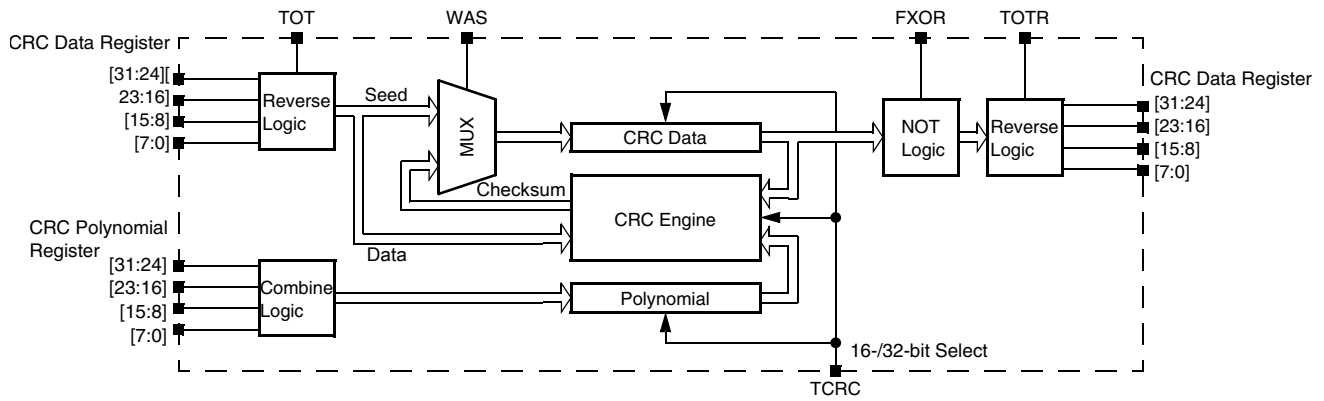


Figure 29-1. Programmable cyclic redundancy check (CRC) block diagram

### 29.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

#### 29.1.3.1 Run mode

This is the basic mode of operation.

#### 29.1.3.2 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.

## 29.2 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	<a href="#">29.2.1/583</a>
4003_2004	CRC Polynomial register (CRC_GPOLY)	32	R/W	0000_1021h	<a href="#">29.2.2/584</a>
4003_2008	CRC Control register (CRC_CTRL)	32	R/W	0000_0000h	<a href="#">29.2.3/584</a>

## 29.2.1 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																																
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	

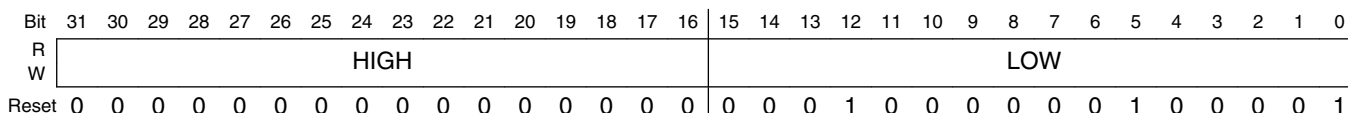
### CRC\_DATA field descriptions

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.
7–0 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.

### 29.2.2 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



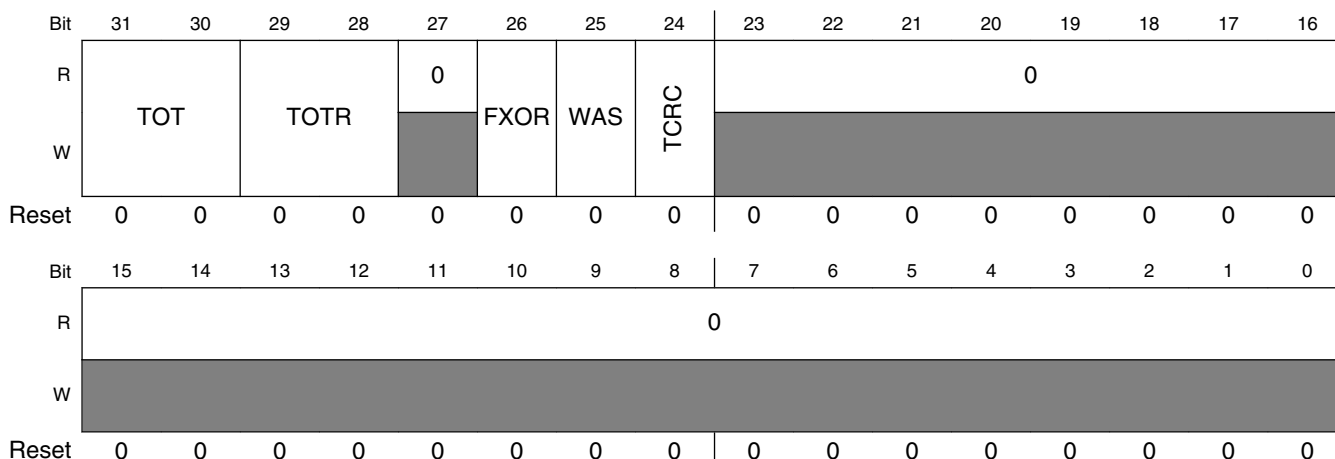
#### CRC\_GPOLY field descriptions

Field	Description
31–16 HIGH	High Polynomial 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).
15–0 LOW	Low Polynomial Half-word  Writable and readable in both 32-bit and 16-bit CRC modes.

### 29.2.3 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	<p>Type Of Transpose For Writes</p> <p>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.</p> <p>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.</p>
29–28 TOTR	<p>Type Of Transpose For Read</p> <p>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.</p> <p>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.</p>
27 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
26 FXOR	<p>Complement Read Of CRC Data Register</p> <p>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.</p> <p>0 No XOR on reading. 1 Invert or complement the read value of the CRC Data register.</p>
25 WAS	<p>Write CRC Data Register As Seed</p> <p>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.</p> <p>0 Writes to the CRC data register are data values. 1 Writes to the CRC data register are seed values.</p>
24 TCRC	<p>Width of CRC protocol.</p> <p>0 16-bit CRC protocol. 1 32-bit CRC protocol.</p>
23–0 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>

## 29.3 Functional description

### 29.3.1 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.

### 29.3.2 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.

#### 29.3.2.1 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.

### 29.3.2.2 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 bitwise, 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.

## 29.3.3 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.

### 29.3.3.1 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.

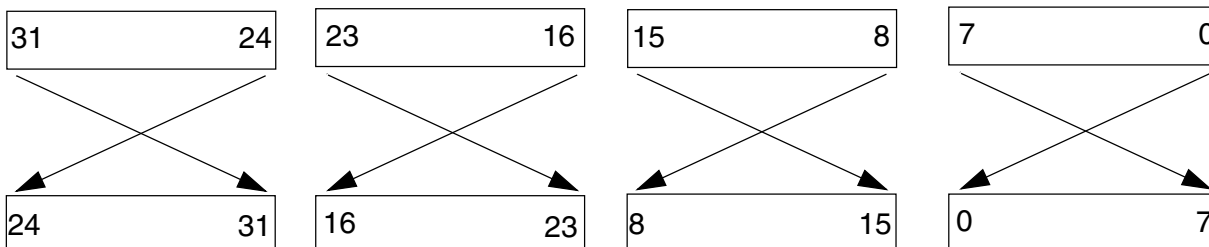
**Functional description**

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]}

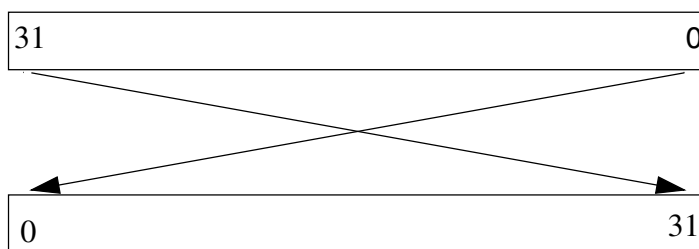


**Figure 29-5. Transpose type 01**

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 29-6. 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]}

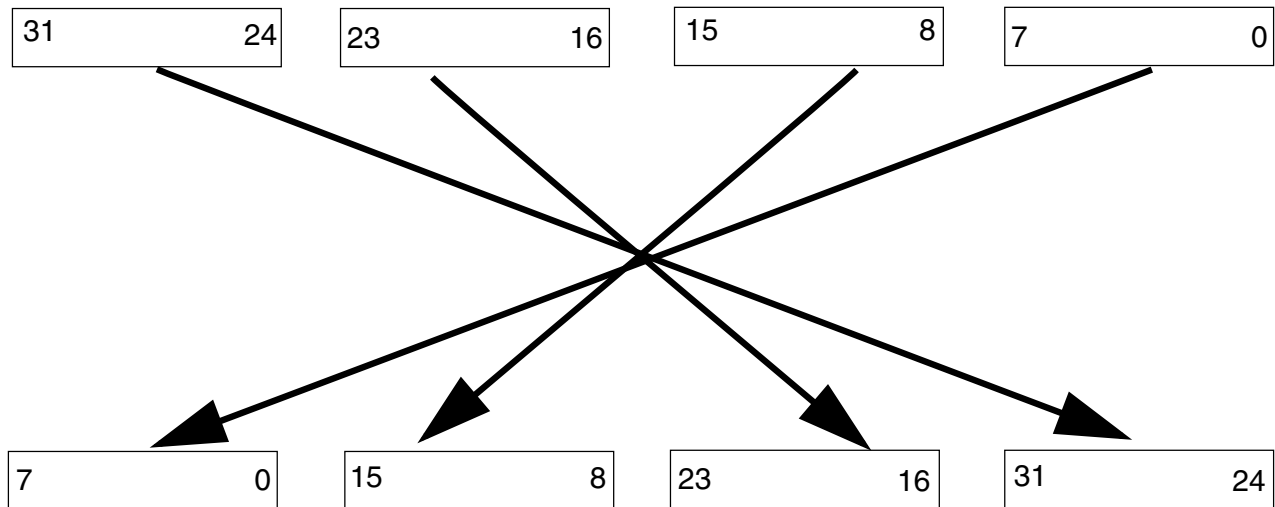


Figure 29-7. Transpose type 11

**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.

**29.3.4 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 30

## Analog-to-Digital Converter (ADC)

### 30.1 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.

#### 30.1.1 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

### 30.1.2 Block diagram

The following figure is the ADC module block diagram.



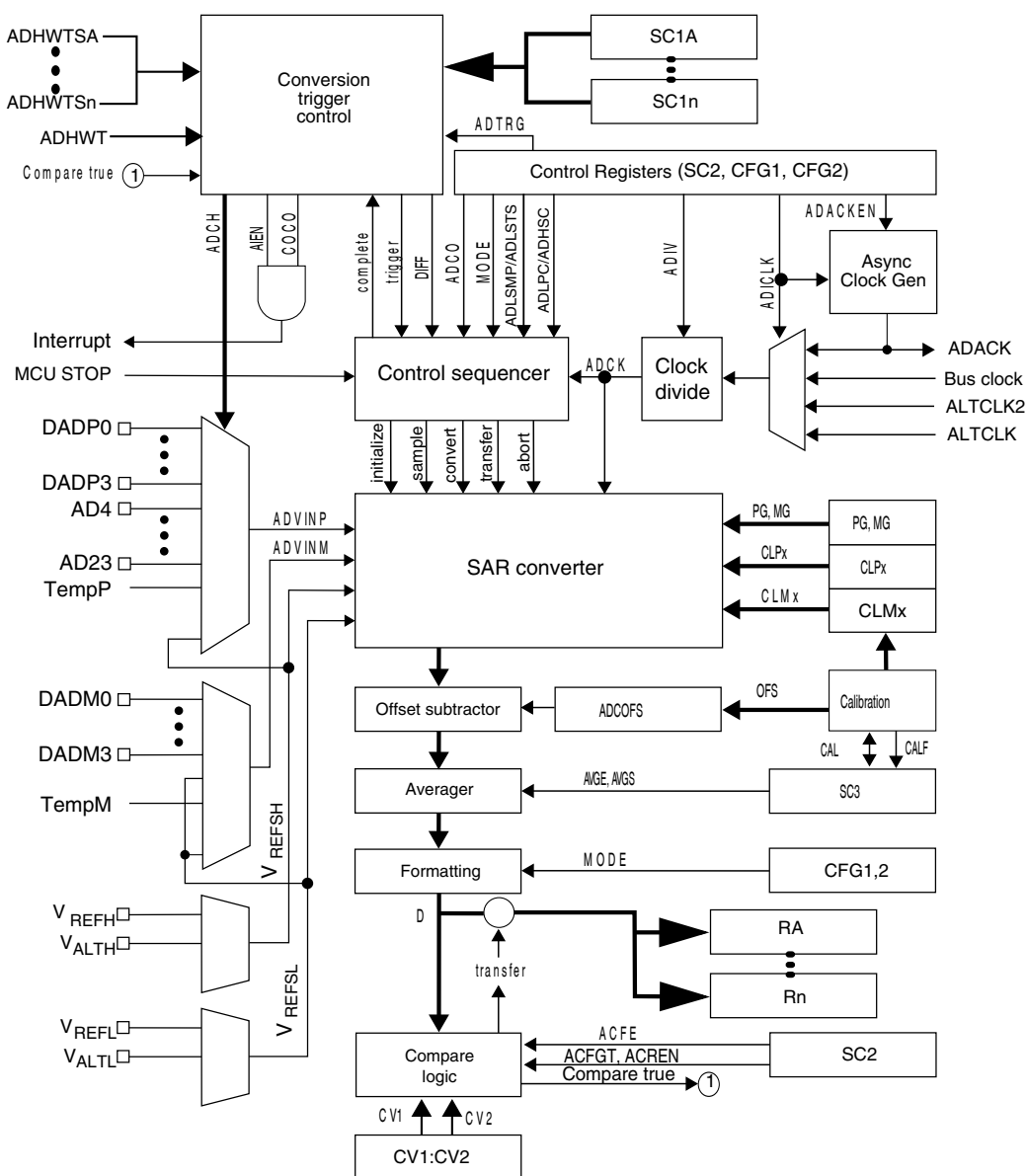


Figure 30-1. ADC block diagram

## 30.2 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

Refer to ADC configuration section in chip configuration chapter for the number of channels supported on this device.

Table 30-1. 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 <sub>REFSH</sub>	Voltage Reference Select High	I
V <sub>REFSL</sub>	Voltage Reference Select Low	I
V <sub>DDA</sub>	Analog Power Supply	I
V <sub>SSA</sub>	Analog Ground	I

### 30.2.1 Analog Power (V<sub>DDA</sub>)

The ADC analog portion uses V<sub>DDA</sub> as its power connection. In some packages, V<sub>DDA</sub> is connected internally to V<sub>DD</sub>. If externally available, connect the V<sub>DDA</sub> pin to the same voltage potential as V<sub>DD</sub>. External filtering may be necessary to ensure clean V<sub>DDA</sub> for good results.

### 30.2.2 Analog Ground (V<sub>SSA</sub>)

The ADC analog portion uses V<sub>SSA</sub> as its ground connection. In some packages, V<sub>SSA</sub> is connected internally to V<sub>SS</sub>. If externally available, connect the V<sub>SSA</sub> pin to the same voltage potential as V<sub>SS</sub>.

### 30.2.3 Voltage Reference Select

V<sub>REFSH</sub> and V<sub>REFSL</sub> 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<sub>REFSH</sub> and V<sub>REFSL</sub>. Each pair contains a positive reference that must be between the minimum Ref Voltage High and V<sub>DDA</sub>, and a ground reference that must be at the same potential as V<sub>SSA</sub>. The two pairs are external (V<sub>REFH</sub> and V<sub>REFL</sub>) and alternate (V<sub>ALTH</sub> and V<sub>ALTTL</sub>). These voltage references are selected using SC2[REFSEL]. The alternate V<sub>ALTH</sub> and V<sub>ALTTL</sub> 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  $V_{SSA}$ .

### 30.2.4 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.

### 30.2.5 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. See the chip configuration chapter for ADC connections specific to this MCU.

## 30.3 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	<a href="#">30.3.1/596</a>
4003_B004	ADC Status and Control Registers 1 (ADC0_SC1B)	32	R/W	0000_001Fh	<a href="#">30.3.1/596</a>
4003_B008	ADC Configuration Register 1 (ADC0_CFG1)	32	R/W	0000_0000h	<a href="#">30.3.2/599</a>
4003_B00C	ADC Configuration Register 2 (ADC0_CFG2)	32	R/W	0000_0000h	<a href="#">30.3.3/601</a>
4003_B010	ADC Data Result Register (ADC0_RA)	32	R	0000_0000h	<a href="#">30.3.4/602</a>
4003_B014	ADC Data Result Register (ADC0_RB)	32	R	0000_0000h	<a href="#">30.3.4/602</a>
4003_B018	Compare Value Registers (ADC0_CV1)	32	R/W	0000_0000h	<a href="#">30.3.5/603</a>
4003_B01C	Compare Value Registers (ADC0_CV2)	32	R/W	0000_0000h	<a href="#">30.3.5/603</a>

Table continues on the next page...

## ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_B020	Status and Control Register 2 (ADC0_SC2)	32	R/W	0000_0000h	<a href="#">30.3.6/604</a>
4003_B024	Status and Control Register 3 (ADC0_SC3)	32	R/W	0000_0000h	<a href="#">30.3.7/606</a>
4003_B028	ADC Offset Correction Register (ADC0_OFS)	32	R/W	0000_0004h	<a href="#">30.3.8/608</a>
4003_B02C	ADC Plus-Side Gain Register (ADC0_PG)	32	R/W	0000_8200h	<a href="#">30.3.9/608</a>
4003_B030	ADC Minus-Side Gain Register (ADC0_MG)	32	R/W	0000_8200h	<a href="#">30.3.10/609</a>
4003_B034	ADC Plus-Side General Calibration Value Register (ADC0_CLPD)	32	R/W	0000_000Ah	<a href="#">30.3.11/609</a>
4003_B038	ADC Plus-Side General Calibration Value Register (ADC0_CLPS)	32	R/W	0000_0020h	<a href="#">30.3.12/610</a>
4003_B03C	ADC Plus-Side General Calibration Value Register (ADC0_CLP4)	32	R/W	0000_0200h	<a href="#">30.3.13/610</a>
4003_B040	ADC Plus-Side General Calibration Value Register (ADC0_CLP3)	32	R/W	0000_0100h	<a href="#">30.3.14/611</a>
4003_B044	ADC Plus-Side General Calibration Value Register (ADC0_CLP2)	32	R/W	0000_0080h	<a href="#">30.3.15/611</a>
4003_B048	ADC Plus-Side General Calibration Value Register (ADC0_CLP1)	32	R/W	0000_0040h	<a href="#">30.3.16/612</a>
4003_B04C	ADC Plus-Side General Calibration Value Register (ADC0_CLP0)	32	R/W	0000_0020h	<a href="#">30.3.17/612</a>
4003_B054	ADC Minus-Side General Calibration Value Register (ADC0_CLMD)	32	R/W	0000_000Ah	<a href="#">30.3.18/613</a>
4003_B058	ADC Minus-Side General Calibration Value Register (ADC0_CLMS)	32	R/W	0000_0020h	<a href="#">30.3.19/613</a>
4003_B05C	ADC Minus-Side General Calibration Value Register (ADC0_CLM4)	32	R/W	0000_0200h	<a href="#">30.3.20/614</a>
4003_B060	ADC Minus-Side General Calibration Value Register (ADC0_CLM3)	32	R/W	0000_0100h	<a href="#">30.3.21/614</a>
4003_B064	ADC Minus-Side General Calibration Value Register (ADC0_CLM2)	32	R/W	0000_0080h	<a href="#">30.3.22/615</a>
4003_B068	ADC Minus-Side General Calibration Value Register (ADC0_CLM1)	32	R/W	0000_0040h	<a href="#">30.3.23/615</a>
4003_B06C	ADC Minus-Side General Calibration Value Register (ADC0_CLM0)	32	R/W	0000_0020h	<a href="#">30.3.24/616</a>

### 30.3.1 ADC Status and Control Registers 1 (ADCx\_SC1n)

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

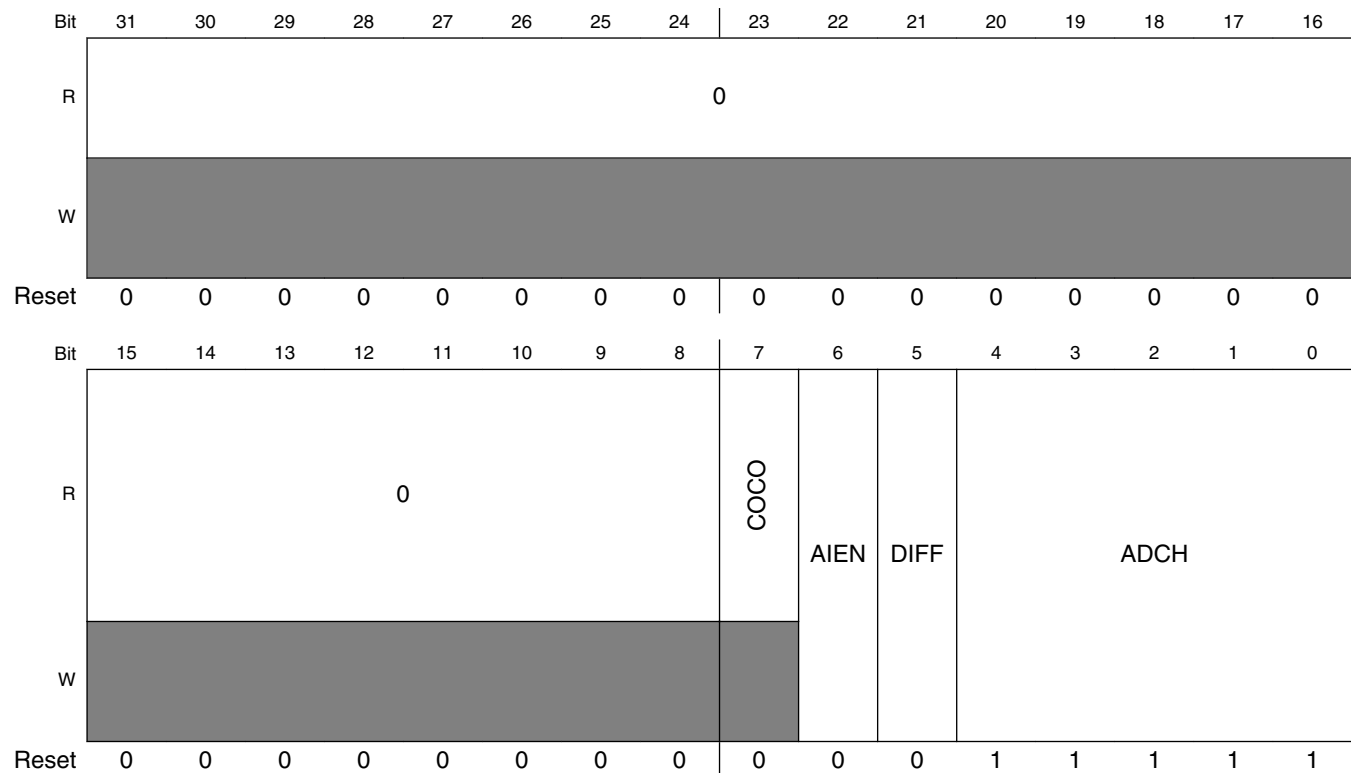
trigger mode. See the chip configuration information about the number of SC1n registers specific to this device. The SC1n registers have identical fields, and are used in a "ping-pong" approach to control ADC operation.

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.

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: 4003\_B000h base + 0h offset + (4d × i), where i=0d to 1d



**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

*Table continues on the next page...*

## ADCx\_SC1n field descriptions (continued)

Field	Description
	<p>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.</p> <p>0 Conversion is not completed. 1 Conversion is completed.</p>
6 AIEN	<p>Interrupt Enable</p> <p>Enables conversion complete interrupts. When COCO becomes set while the respective AIEN is high, an interrupt is asserted.</p> <p>0 Conversion complete interrupt is disabled. 1 Conversion complete interrupt is enabled.</p>
5 DIFF	<p>Differential Mode Enable</p> <p>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.</p> <p>0 Single-ended conversions and input channels are selected. 1 Differential conversions and input channels are selected.</p>
4–0 ADCH	<p>Input channel select</p> <p>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.</p> <p><b>NOTE:</b> 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.</p> <p>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.</p> <p>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. 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.</p>

Table continues on the next page...

## ADCx\_SC1n field descriptions (continued)

Field	Description
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 <sub>REFSH</sub> is selected as input; when DIFF=1, -V <sub>REFSH</sub> (differential) is selected as input. Voltage reference selected is determined by SC2[REFSEL].
11110	When DIFF=0, V <sub>REFSL</sub> is selected as input; when DIFF=1, it is reserved. Voltage reference selected is determined by SC2[REFSEL].
11111	Module is disabled.

## 30.3.2 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: 4003\_B000h base + 8h offset = 4003\_B008h

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								ADLPC	ADIV		ADLSMP	MODE		ADICLK	
W									ADLPC	ADIV		ADLSMP	MODE		ADICLK	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## 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.  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
1–0 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 Alternate clock 2 (ALTCLK2) 10 Alternate clock (ALTCLK) 11 Asynchronous clock (ADACK)



### 30.3.3 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: 4003\_B000h base + Ch offset = 4003\_B00Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
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	0								0			MUXSEL	ADACKEN	ADHSC	ADLSTS	
W	[Reserved]								[Reserved]			MUXSEL	ADACKEN	ADHSC	ADLSTS	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

#### 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 MUXSEL	ADC Mux Select  Changes the ADC mux setting to select between alternate sets of ADC channels.  0 ADxxa channels are selected. 1 ADxxb channels are selected.
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.  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

Table continues on the next page...

**ADCx\_CFG2 field descriptions (continued)**

Field	Description
	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.
1-0 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.

**30.3.4 ADC Data Result Register (ADCx\_Rn)**

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 30-43. 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
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

*Table continues on the next page...*

**Table 30-43. Data result register description (continued)**

Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format
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: 4003\_B000h base + 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	0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**ADCx\_Rn field descriptions**

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–0 D	Data result

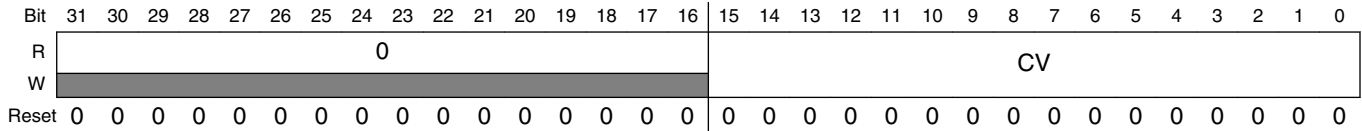
**30.3.5 Compare Value Registers (ADCx\_CVn)**

The Compare Value Registers (CV1 and CV2) contain a compare value used to compare the conversion result when the compare function is enabled, that is, SC2[ACFE]=1. This register is formatted in the same way as the Rn registers in different modes of operation for both bit position definition and value format using unsigned or sign-extended 2's complement. Therefore, the compare function uses only the CVn fields that are related to the ADC mode of operation.

The compare value 2 register (CV2) is used only when the compare range function is enabled, that is, SC2[ACREN]=1.

## Memory map and register definitions

Address: 4003\_B000h base + 18h offset + (4d × i), where i=0d to 1d



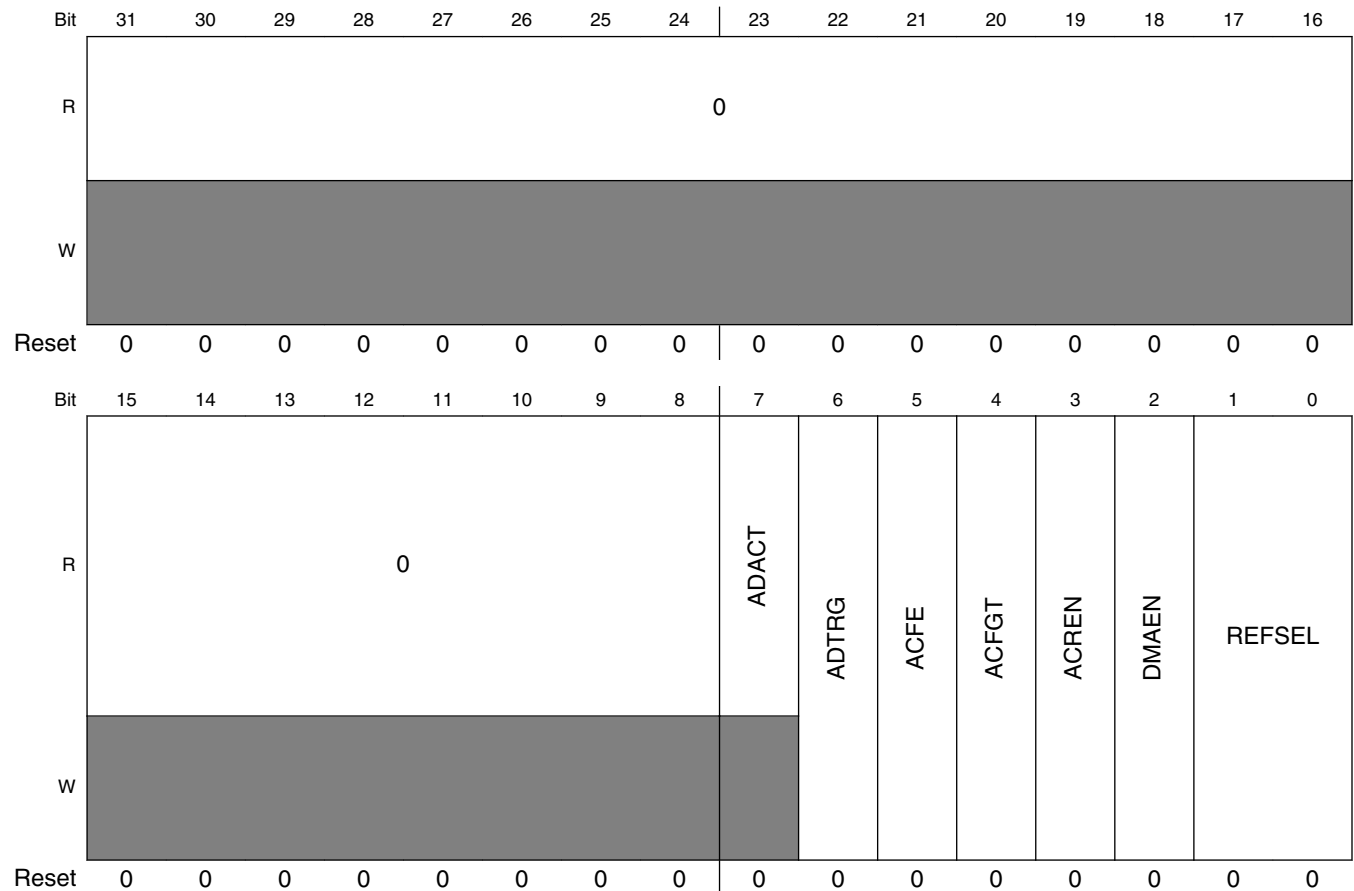
### ADCx\_CVn field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–0 CV	Compare Value.

## 30.3.6 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: 4003\_B000h base + 20h offset = 4003\_B020h



## 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: <ul style="list-style-type: none"> <li>• Software trigger: When software trigger is selected, a conversion is initiated following a write to SC1A.</li> <li>• 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.</li> </ul> 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.
1–0 REFSEL	Voltage Reference Selection  Selects the voltage reference source used for conversions.  00 Default voltage reference pin pair, that is, external pins V <sub>REFH</sub> and V <sub>REFL</sub>

*Table continues on the next page...*

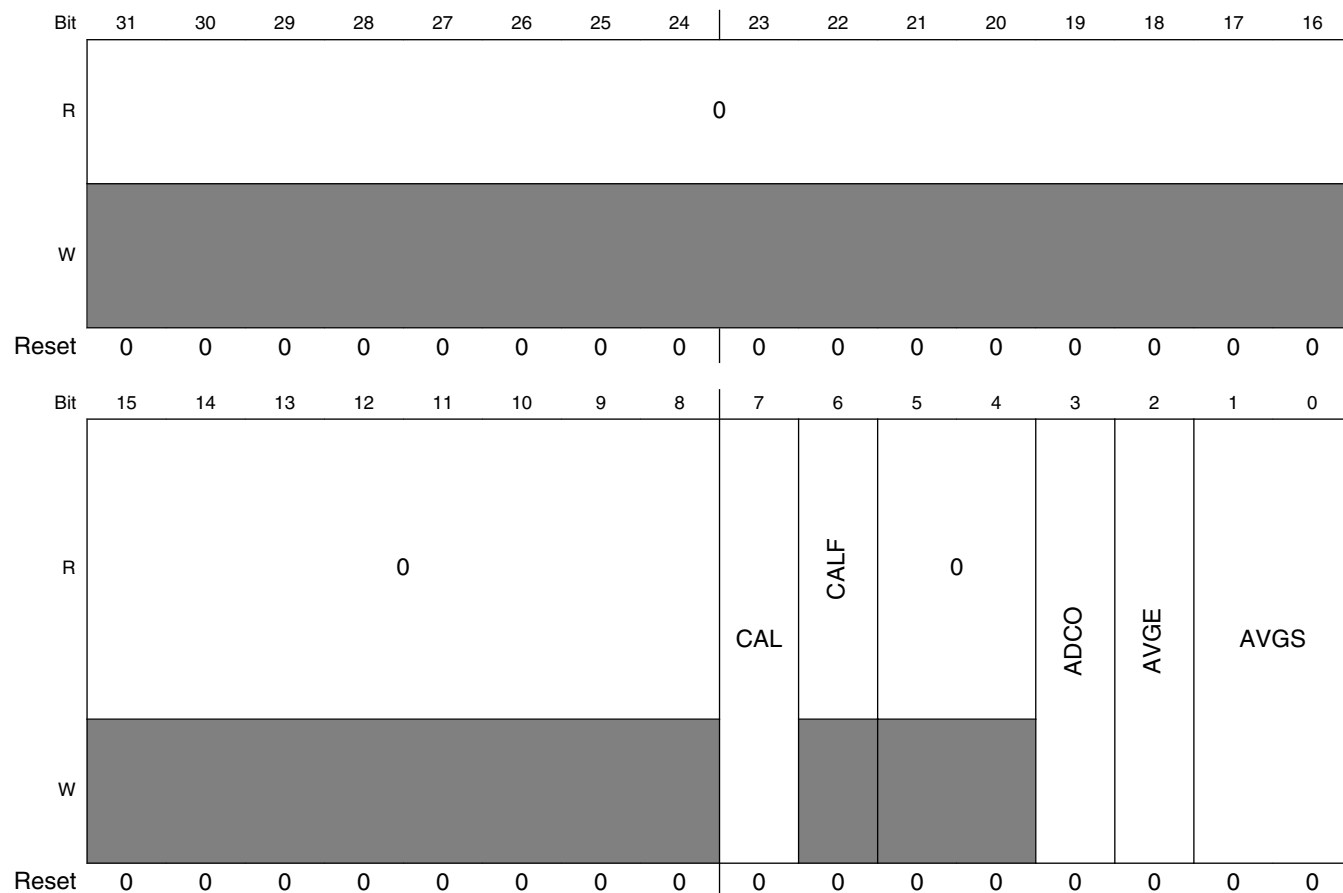
**ADCx\_SC2 field descriptions (continued)**

Field	Description
01	Alternate reference pair, that is, V <sub>ALTH</sub> and V <sub>ALTL</sub> . 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

**30.3.7 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: 4003\_B000h base + 24h offset = 4003\_B024h



**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.

Table continues on the next page...

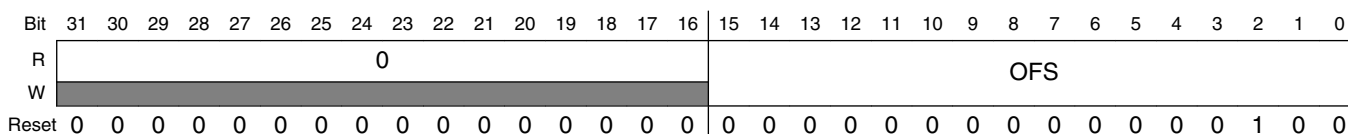
## ADCx\_SC3 field descriptions (continued)

Field	Description
7 CAL	<p>Calibration</p> <p>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.</p>
6 CALF	<p>Calibration Failed Flag</p> <p>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.</p> <p>0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.</p>
5–4 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
3 ADCO	<p>Continuous Conversion Enable</p> <p>Enables continuous conversions.</p> <p>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.</p>
2 AVGE	<p>Hardware Average Enable</p> <p>Enables the hardware average function of the ADC.</p> <p>0 Hardware average function disabled. 1 Hardware average function enabled.</p>
1–0 AVGS	<p>Hardware Average Select</p> <p>Determines how many ADC conversions will be averaged to create the ADC average result.</p> <p>00 4 samples averaged. 01 8 samples averaged. 10 16 samples averaged. 11 32 samples averaged.</p>

### 30.3.8 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.

Address: 4003\_B000h base + 28h offset = 4003\_B028h



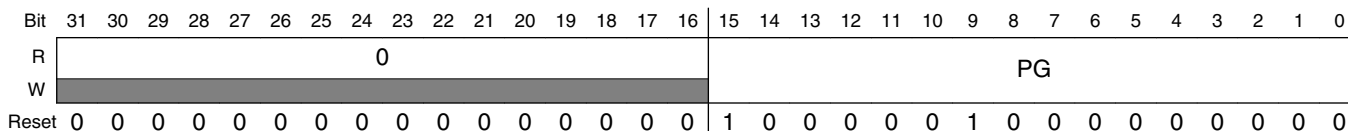
#### ADCx\_OFS field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–0 OFS	Offset Error Correction Value

### 30.3.9 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 ADPG15 and ADPG14. This register must be written by the user with the value described in the calibration procedure. Otherwise, the gain error specifications may not be met.

Address: 4003\_B000h base + 2Ch offset = 4003\_B02Ch





**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.
15–0 PG	Plus-Side Gain

**30.3.10 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 ADMG15 and ADMG14. This register must be written by the user with the value described in the calibration procedure. Otherwise, the gain error specifications may not be met.

Address: 4003\_B000h base + 30h offset = 4003\_B030h

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	0																0															
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.
15–0 MG	Minus-Side Gain

**30.3.11 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.

## Memory map and register definitions

Address: 4003\_B000h base + 34h offset = 4003\_B034h

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	0																0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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.
5–0 CLPD	Calibration Value Calibration Value

## 30.3.12 ADC Plus-Side General Calibration Value Register (ADCx\_CLPS)

For more information, see CLPD register description.

Address: 4003\_B000h base + 38h offset = 4003\_B038h

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	0																0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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.
5–0 CLPS	Calibration Value Calibration Value

## 30.3.13 ADC Plus-Side General Calibration Value Register (ADCx\_CLP4)

For more information, see CLPD register description.

Address: 4003\_B000h base + 3Ch offset = 4003\_B03Ch

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	0																0															
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.
9–0 CLP4	Calibration Value Calibration Value

**30.3.14 ADC Plus-Side General Calibration Value Register (ADCx\_CLP3)**

For more information, see CLPD register description.

Address: 4003\_B000h base + 40h offset = 4003\_B040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																CLP3															
W	0																1															
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\_CLP3 field descriptions**

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–0 CLP3	Calibration Value Calibration Value

**30.3.15 ADC Plus-Side General Calibration Value Register (ADCx\_CLP2)**

For more information, see CLPD register description.

Address: 4003\_B000h base + 44h offset = 4003\_B044h

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																CLP2															
W	0																1															
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\_CLP2 field descriptions**

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

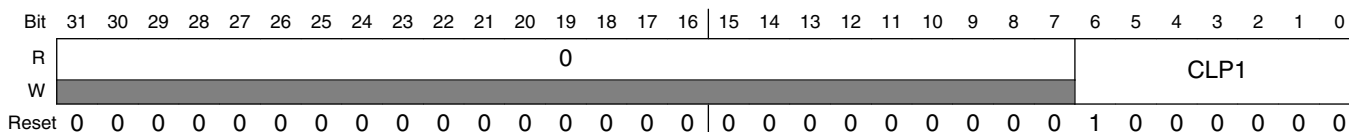
**ADCx\_CLP2 field descriptions (continued)**

Field	Description
7–0 CLP2	Calibration Value
	Calibration Value

**30.3.16 ADC Plus-Side General Calibration Value Register (ADCx\_CLP1)**

For more information, see CLPD register description.

Address: 4003\_B000h base + 48h offset = 4003\_B048h



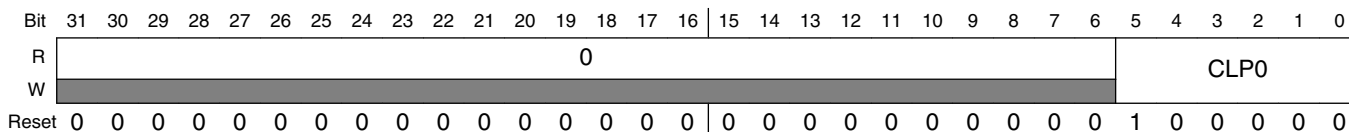
**ADCx\_CLP1 field descriptions**

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–0 CLP1	Calibration Value
	Calibration Value

**30.3.17 ADC Plus-Side General Calibration Value Register (ADCx\_CLP0)**

For more information, see CLPD register description.

Address: 4003\_B000h base + 4Ch offset = 4003\_B04Ch



**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.
5–0 CLP0	Calibration Value
	Calibration Value

### 30.3.18 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.

Address: 4003\_B000h base + 54h offset = 4003\_B054h

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.
5–0 CLMD	Calibration Value Calibration Value

### 30.3.19 ADC Minus-Side General Calibration Value Register (ADCx\_CLMS)

For more information, see CLMD register description.

Address: 4003\_B000h base + 58h offset = 4003\_B058h

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

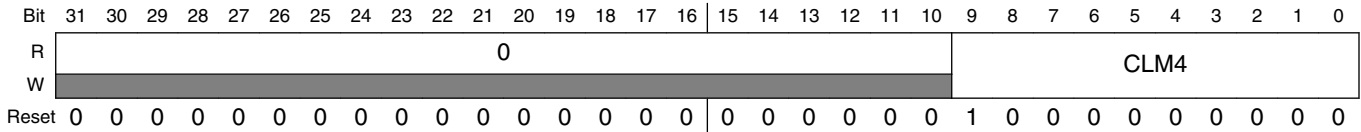
#### ADCx\_CLMS field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–0 CLMS	Calibration Value Calibration Value

### 30.3.20 ADC Minus-Side General Calibration Value Register (ADCx\_CLM4)

For more information, see CLMD register description.

Address: 4003\_B000h base + 5Ch offset = 4003\_B05Ch



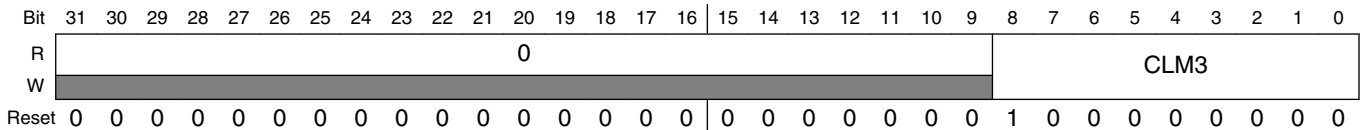
#### ADCx\_CLM4 field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–0 CLM4	Calibration Value Calibration Value

### 30.3.21 ADC Minus-Side General Calibration Value Register (ADCx\_CLM3)

For more information, see CLMD register description.

Address: 4003\_B000h base + 60h offset = 4003\_B060h



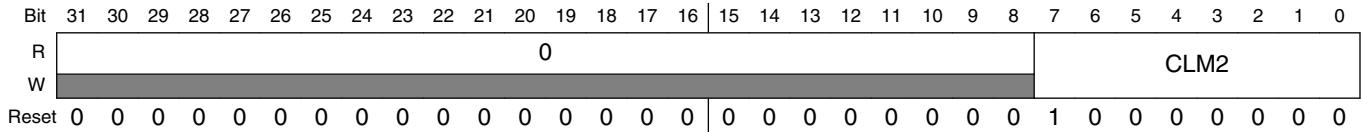
#### ADCx\_CLM3 field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–0 CLM3	Calibration Value Calibration Value

### 30.3.22 ADC Minus-Side General Calibration Value Register (ADCx\_CLM2)

For more information, see CLMD register description.

Address: 4003\_B000h base + 64h offset = 4003\_B064h



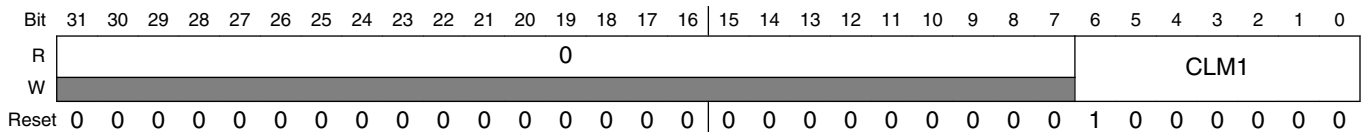
#### ADCx\_CLM2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–0 CLM2	Calibration Value Calibration Value

### 30.3.23 ADC Minus-Side General Calibration Value Register (ADCx\_CLM1)

For more information, see CLMD register description.

Address: 4003\_B000h base + 68h offset = 4003\_B068h



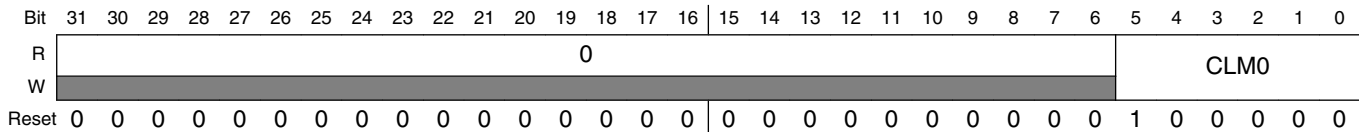
#### ADCx\_CLM1 field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–0 CLM1	Calibration Value Calibration Value

### 30.3.24 ADC Minus-Side General Calibration Value Register (ADCx\_CLM0)

For more information, see CLMD register description.

Address: 4003\_B000h base + 6Ch offset = 4003\_B06Ch



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.
5–0 CLM0	Calibration Value Calibration Value

## 30.4 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.

## 30.4.1 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.
- ALTCLK2: As defined for this MCU. See the chip configuration information. Conversions are possible using ALTCLK2 as the input clock source while the MCU is in Normal Stop mode.
- 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  $\mu$ s 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.

### 30.4.2 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.

### 30.4.3 Hardware trigger and channel selects

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when  $SC2[ADTRG]$  is set and a hardware trigger select event,  $ADHWTSn$ , has occurred.

This source is not available on all MCUs. See the Chip Configuration chapter for information on the ADHWT source and the  $ADHWTSn$  configurations specific to this MCU.

When an ADHWT source is available and hardware trigger is enabled, that is  $SC2[ADTRG]=1$ , a conversion is initiated on the rising-edge of ADHWT after a hardware trigger select event, that is,  $ADHWTSn$ , has occurred. If a conversion is in progress when a rising-edge of a trigger occurs, the rising-edge is ignored. In continuous convert configuration, only the initial rising-edge to launch continuous conversions is observed, and until conversion is aborted, the ADC continues to do conversions on the same  $SCn$  register that initiated the conversion. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

The hardware trigger select event,  $ADHWTSn$ , must be set prior to the receipt of the ADHWT signal. If these conditions are not met, the converter may ignore the trigger or use the incorrect configuration. If a hardware trigger select event is asserted during a conversion, it must stay asserted until the end of current conversion and remain set until the receipt of the ADHWT signal to trigger a new conversion. The channel and status fields selected for the conversion depend on the active trigger select signal:

- $ADHWTS_A$  active selects  $SC1A$ .
- $ADHWTS_n$  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.

## 30.4.4 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

### 30.4.4.1 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.

## Functional description

- 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.

### 30.4.4.2 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.

### 30.4.4.3 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.

### 30.4.4.4 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  $f_{ADCK}$ .

### 30.4.4.5 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.

$$\text{ConversionTime} = \text{SFCAdder} + \text{AverageNum} \times (\text{BCT} + \text{LSTAdder} + \text{HSCAdder})$$

**Figure 30-62. Conversion time equation**

**Table 30-70. 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 <sup>1</sup>
1	0	11	5 $\mu$ s + 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 <sup>1</sup>
0	0	11	5 $\mu$ s + 5 ADCK cycles + 5 bus clock cycles

1. To achieve this time,  $CFG2[ADACKEN]$  must be 1 for at least 5  $\mu$ s prior to the conversion is initiated.

**Table 30-71. 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 30-72. 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 30-72. Base conversion time (BCT) (continued)**

Mode	Base conversion time (BCT)
16b single-ended	25 ADCK cycles
16b differential	34 ADCK cycles

**Table 30-73. 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 30-74. 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.

## 30.4.4.6 Conversion time examples

The following examples use the [Figure 30-62](#), and the information provided in [Table 30-70](#) through [Table 30-74](#).

### 30.4.4.6.1 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 [Figure 30-62](#), and the information provided in [Table 30-70](#) through [Table 30-74](#). The table below lists the variables of [Figure 30-62](#).

**Table 30-75. 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  $\mu$ s.

#### 30.4.4.6.2 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 [Figure 30-62](#), and the information provided in [Table 30-70](#) through [Table 30-74](#). The following table lists the variables of the [Figure 30-62](#).

**Table 30-76. 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  $\mu$ s, that is, AverageNum. This results in a total conversion time of 1.844 ms.

### 30.4.4.6.3 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 [Figure 30-62](#), and the information provided in [Table 30-70](#) through [Table 30-74](#). The table below lists the variables of [Figure 30-62](#).

**Table 30-77. 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 the preceding table. Therefore, for bus clock and ADCK frequency equal to 20 MHz, the resulting conversion time is 1.45  $\mu$ s.

### 30.4.4.7 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.

## 30.4.5 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 30-78. 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 <b>Or</b> the result is greater than CV2.
0	1	Greater than	Inside range, not inclusive	Compare true if the result is less than CV1 <b>And</b> 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 <b>And</b> 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 <b>Or</b> 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.

## **30.4.6 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](#).

### **30.4.7 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.

### 30.4.8 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.

$$\text{Temp} = 25 - \left( \left( V_{\text{TEMP}} - V_{\text{TEMP25}} \right) \div m \right)$$

**Figure 30-63. Approximate transfer function of the temperature sensor**

where:

- $V_{\text{TEMP}}$  is the voltage of the temperature sensor channel at the ambient temperature.
- $V_{\text{TEMP25}}$  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  $V_{\text{TEMP25}}$  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).

### **30.4.9 MCU wait mode operation**

Wait mode is a lower-power consumption Standby mode from which recovery is fast because the clock sources remain active.

If a conversion is in progress when the MCU enters Wait mode, it continues until completion. Conversions can be initiated while the MCU is in Wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, ADACK, and Alternate Clock sources 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.

### **30.4.10 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.



### 30.4.10.1 Normal Stop mode with Bus Clock selected

If the Bus Clock is 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.

### 30.4.10.2 Normal Stop mode with ADACK or Alternate clock sources enabled

If ADACK or an Alternate clock source is selected as the conversion clock, the ADC continues operation during Normal Stop mode. See the chip configuration chapter for configuration information for this MCU.

If a conversion is in progress when the MCU enters Normal Stop mode, it continues until completion. Conversions can be initiated while the MCU is in Normal Stop mode by means of the hardware trigger or if continuous conversions are enabled.

If the compare and hardware averaging functions are disabled, a conversion complete event sets SC1n[COCO] and generates an ADC interrupt to wake the MCU from Normal Stop mode if the respective ADC interrupt is enabled, that is, when SC1n[AIEN]=1. The result register, Rn, will contain the data from the first completed conversion that occurred during Normal Stop mode. If the hardware averaging function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, when the selected number of conversions are completed. If the compare function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, only if the compare conditions are met. If a single conversion is selected and the compare is not true, the ADC will return to its Idle state and cannot wake the MCU from Normal Stop mode unless a new conversion is initiated by another hardware trigger.

## 30.4.11 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.

## 30.5 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 30-73](#), [Table 30-74](#), and [Table 30-75](#).

### Note

Hexadecimal values are designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

### 30.5.1 ADC module initialization example

#### 30.5.1.1 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.

### 30.5.1.2 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 V <sub>REFH</sub> and V <sub>REFL</sub> ).

#### 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.

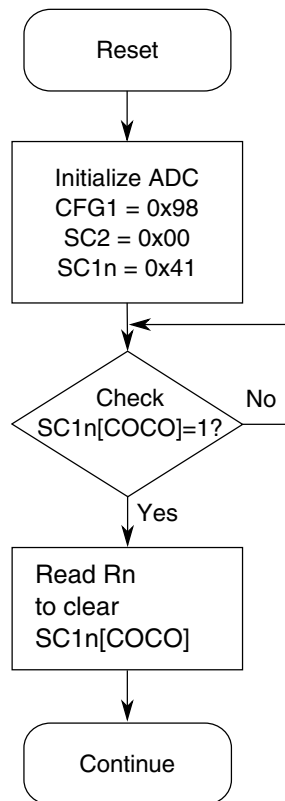


Figure 30-64. Initialization flowchart example

## 30.6 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](#).

### 30.6.1 External pins and routing

#### 30.6.1.1 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.

### 30.6.1.2 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  $V_{ALTLL}$ . These voltage references are selected using  $SC2[REFSEL]$ . The alternate voltage reference pair,  $V_{ALTH}$  and  $V_{ALTLL}$ , 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  $\mu\text{F}$  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.

### 30.6.1.3 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  $\mu\text{F}$  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.

## 30.6.2 Sources of error

### 30.6.2.1 Sampling error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy.

$$RAS + RADIN = SC / (FMAX * NUMTAU * CADIN)$$

**Figure 30-65. Sampling equation**

Where:

RAS = External analog source resistance

SC = Number of ADCK cycles used during sample window

CADIN = Internal ADC input capacitance

NUMTAU =  $-\ln(\text{LSBERR} / 2^N)$

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.

### 30.6.2.2 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  $V_{REFH} / (4 \times I_{LEAK} \times 2^N)$  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.

### 30.6.2.3 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  $\mu\text{F}$  low-ESR capacitor from  $V_{REFH}$  to  $V_{REFL}$ .
- There is a 0.1  $\mu\text{F}$  low-ESR capacitor from  $V_{DDA}$  to  $V_{SSA}$ .
- If inductive isolation is used from the primary supply, an additional 1  $\mu\text{F}$  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 or an Alternate clock 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  $\mu\text{F}$  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.

#### 30.6.2.4 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:

$$1\text{LSB} = (V_{REFH}) / 2^N$$

**Figure 30-66. 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  $\pm 1/2$  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.

### 30.6.2.5 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.

### 30.6.2.6 Code jitter, non-monotonicity, and missing codes

Analog-to-digital converters are susceptible to three special forms of error:

- Code jitter: Code jitter is when, at certain points, a given input voltage converts to one of the two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code, and vice-versa. However, even small amounts of system noise can cause the converter to be indeterminate, between two codes, for a range of input voltages around the transition voltage.

## Application information

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally, the techniques discussed in [Noise-induced errors](#) reduces this error.

- Non-monotonicity: Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage.
- Missing codes: Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

# Chapter 31

## Comparator (CMP)

### 31.1 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.

#### 31.1.1 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 except VLLS0
- The window and filter functions are not available in the following modes:
  - Stop
  - VLPS
  - LLS
  - VLLSx

### 31.1.2 6-bit DAC key features

- 6-bit resolution
- Selectable supply reference source
- Power Down mode to conserve power when not in use
- Option to route the output to internal comparator input

### 31.1.3 ANMUX key features

- Two 8-to-1 channel mux
- Operational over the entire supply range

### 31.1.4 CMP, DAC and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.

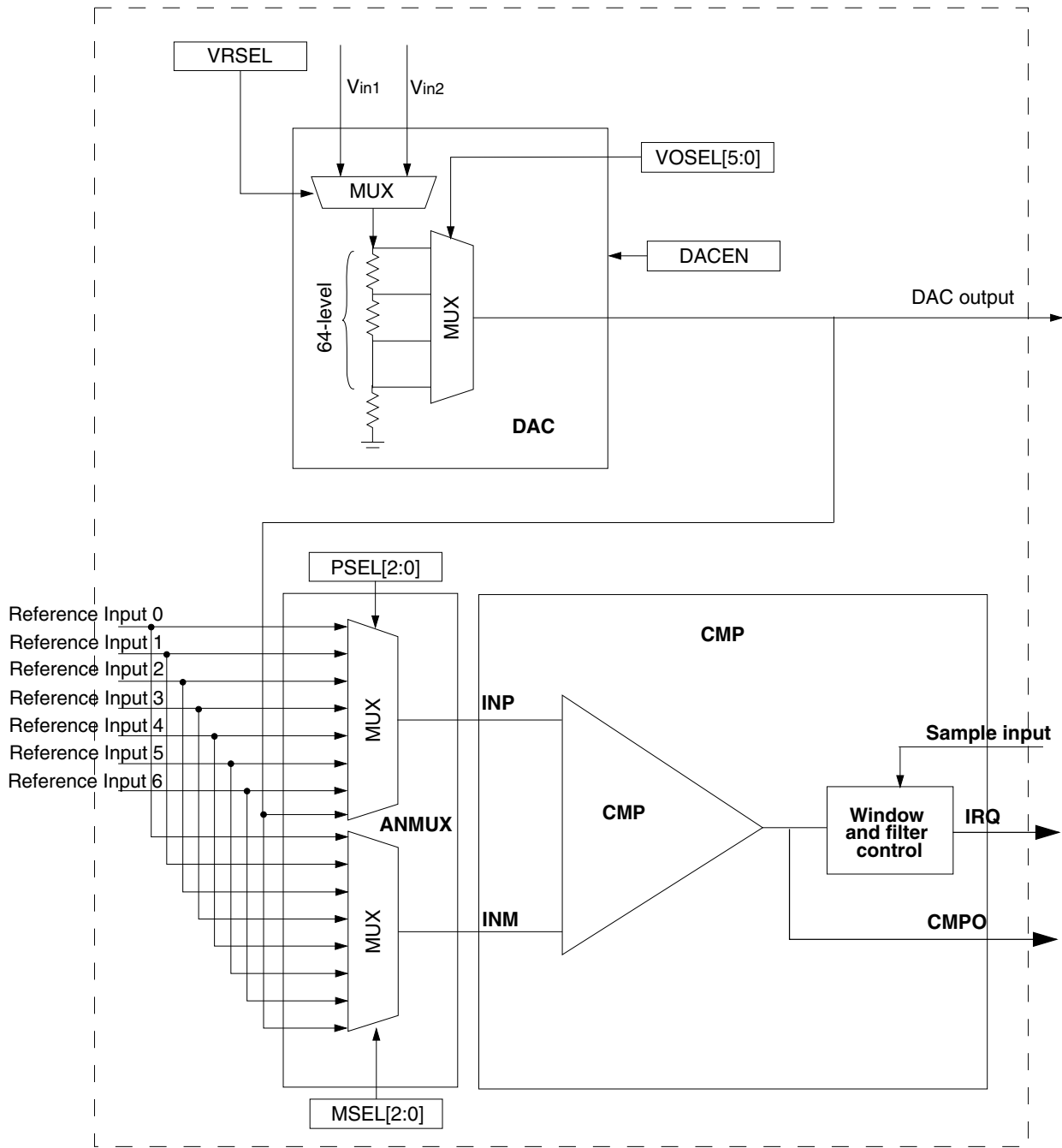
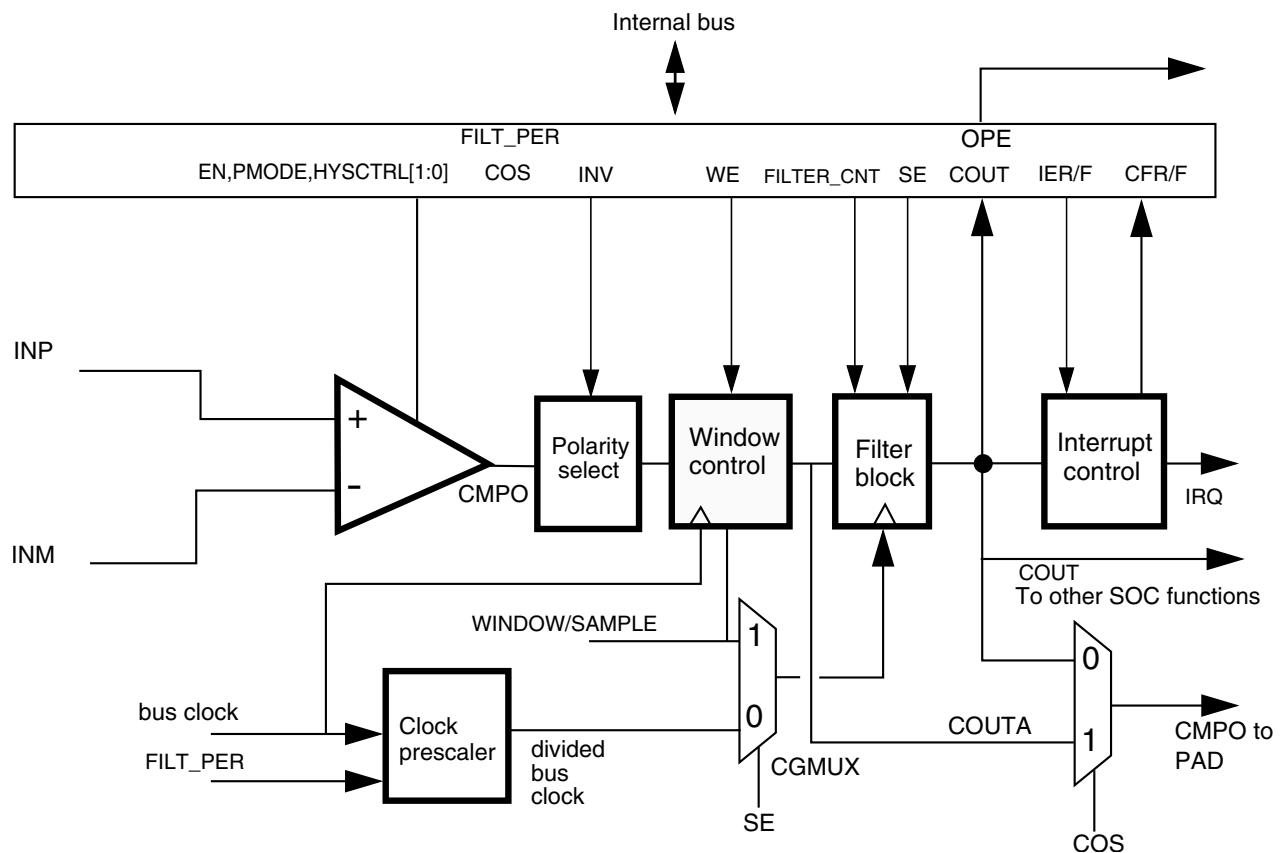


Figure 31-1. CMP, DAC and ANMUX block diagram

### 31.1.5 CMP block diagram

The following figure shows the block diagram for the CMP module.



**Figure 31-2. 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

## Memory map/register definitions

- 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.

## 31.2 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	<a href="#">31.2.1/648</a>
4007_3001	CMP Control Register 1 (CMP0_CR1)	8	R/W	00h	<a href="#">31.2.2/649</a>
4007_3002	CMP Filter Period Register (CMP0_FPR)	8	R/W	00h	<a href="#">31.2.3/651</a>
4007_3003	CMP Status and Control Register (CMP0_SCR)	8	R/W	00h	<a href="#">31.2.4/651</a>
4007_3004	DAC Control Register (CMP0_DACCR)	8	R/W	00h	<a href="#">31.2.5/652</a>
4007_3005	MUX Control Register (CMP0_MUXCR)	8	R/W	00h	<a href="#">31.2.6/653</a>
4007_3008	CMP Control Register 0 (CMP1_CR0)	8	R/W	00h	<a href="#">31.2.1/648</a>
4007_3009	CMP Control Register 1 (CMP1_CR1)	8	R/W	00h	<a href="#">31.2.2/649</a>
4007_300A	CMP Filter Period Register (CMP1_FPR)	8	R/W	00h	<a href="#">31.2.3/651</a>
4007_300B	CMP Status and Control Register (CMP1_SCR)	8	R/W	00h	<a href="#">31.2.4/651</a>
4007_300C	DAC Control Register (CMP1_DACCR)	8	R/W	00h	<a href="#">31.2.5/652</a>
4007_300D	MUX Control Register (CMP1_MUXCR)	8	R/W	00h	<a href="#">31.2.6/653</a>

### 31.2.1 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 output filter accepting a new output state. For information regarding filter programming and latency, see the <a href="#">Functional description</a> .  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.
1-0 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

## 31.2.2 CMP Control Register 1 (CMPx\_CR1)

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read	SE	WE	TRIGM	PMODE	INV	COS	OPE	EN
Write								
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. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations.  0 Sampling mode is not selected. 1 Sampling mode is selected.
6 WE	Windowing Enable

Table continues on the next page...

**CMPx\_CR1 field descriptions (continued)**

Field	Description
	<p>At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations.</p> <p>0 Windowing mode is not selected. 1 Windowing mode is selected.</p>
5 TRIGM	<p>Trigger Mode Enable</p> <p>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.</p> <p>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.</p> <p>Upon setting TRIGM, the CMP and DAC are placed in a standby state until an external timer resource trigger is received.</p> <p>See the chip configuration for details about the external timer resource.</p> <p>0 Trigger mode is disabled. 1 Trigger mode is enabled.</p>
4 PMODE	<p>Power Mode Select</p> <p>See the electrical specifications table in the device Data Sheet for details.</p> <p>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.</p>
3 INV	<p>Comparator INVERT</p> <p>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.</p> <p>0 Does not invert the comparator output. 1 Inverts the comparator output.</p>
2 COS	<p>Comparator Output Select</p> <p>0 Set the filtered comparator output (CMPO) to equal COUT. 1 Set the unfiltered comparator output (CMPO) to equal COUTA.</p>
1 OPE	<p>Comparator Output Pin Enable</p> <p>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.</p> <p>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.</p>
0 EN	<p>Comparator Module Enable</p> <p>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.</p>

*Table continues on the next page...*

**CMPx\_CR1 field descriptions (continued)**

Field	Description
0	Analog Comparator is disabled.
1	Analog Comparator is enabled.

**31.2.3 CMP Filter Period Register (CMPx\_FPR)**

Address: Base address + 2h offset

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

**CMPx\_FPR field descriptions**

Field	Description
7-0 FILT_PER	<p>Filter Sample Period</p> <p>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 <a href="#">Functional description</a>.</p> <p>This field has no effect when CR1[SE]=1. In that case, the external SAMPLE signal is used to determine the sampling period.</p>

**31.2.4 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	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
6 DMAEN	<p>DMA Enable Control</p> <p>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.</p> <p>0 DMA is disabled.</p> <p>1 DMA is enabled.</p>

*Table continues on the next page...*

**CMPx\_SCR field descriptions (continued)**

Field	Description
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 IER	Comparator Interrupt Enable Rising  Enables the CFR interrupt from the CMP. When this field is set, an interrupt will be asserted when CFR is set.  0 Interrupt is disabled. 1 Interrupt is enabled.
3 IEF	Comparator Interrupt Enable Falling  Enables the CFF interrupt from the CMP. When this field is set, an interrupt will be asserted when CFF is set.  0 Interrupt is disabled. 1 Interrupt is enabled.
2 CFR	Analog Comparator Flag Rising  Detects a rising-edge on COUT, when set, during normal operation. CFR is cleared by writing 1 to it. During Stop modes, CFR is level 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 level 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.

**31.2.5 DAC Control Register (CMPx\_DACCR)**

Address: Base address + 4h offset

Bit	7	6	5	4	3	2	1	0
Read	DACEN	VRSEL	VOSEL					
Write								
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.
5–0 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 $V_{in}$ .

## 31.2.6 MUX Control Register (CMPx\_MUXCR)

Address: Base address + 5h offset

Bit	7	6	5	4	3	2	1	0
Read	Reserved	0	PSEL			MSEL		
Write								
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. <b>NOTE:</b> 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
2–0 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
	<b>NOTE:</b> 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

### 31.3 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].

#### 31.3.1 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 31-22. Comparator sample/filter controls**

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation
1	0	X	X	X	X	<b>Disabled</b> See the <a href="#">Disabled mode (# 1)</a> .
2A	1	0	0	0x00	X	<b>Continuous Mode</b> See the <a href="#">Continuous mode (#s 2A &amp; 2B)</a> .
2B	1	0	0	X	0x00	
3A	1	0	1	0x01	X	<b>Sampled, Non-Filtered mode</b> See the <a href="#">Sampled, Non-Filtered mode (#s 3A &amp; 3B)</a> .
3B	1	0	0	0x01	> 0x00	
4A	1	0	1	> 0x01	X	<b>Sampled, Filtered mode</b> See the <a href="#">Sampled, Filtered mode (#s 4A &amp; 4B)</a> .
4B	1	0	0	> 0x01	> 0x00	
5A	1	1	0	0x00	X	<b>Windowed mode</b> Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA. See the <a href="#">Windowed mode (#s 5A &amp; 5B)</a> .
5B	1	1	0	X	0x00	
6	1	1	0	0x01	0x01–0xFF	<b>Windowed/Resampled mode</b> 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 <a href="#">Windowed/Resampled mode (# 6)</a> .
7	1	1	0	> 0x01	0x01–0xFF	<b>Windowed/Filtered mode</b> 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 <a href="#">Windowed/Filtered mode (#7)</a> .
All other combinations of CR1[EN], CR1[WE], CR1[SE], CR0[FILTER_CNT], and FPR[FILT_PER] are illegal.						

## Functional description

For cases where a comparator is used to drive a fault input, for example, for a motor-control module such as FTM, 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.

#### 31.3.1.1 Disabled mode (# 1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

#### 31.3.1.2 Continuous mode (#s 2A & 2B)

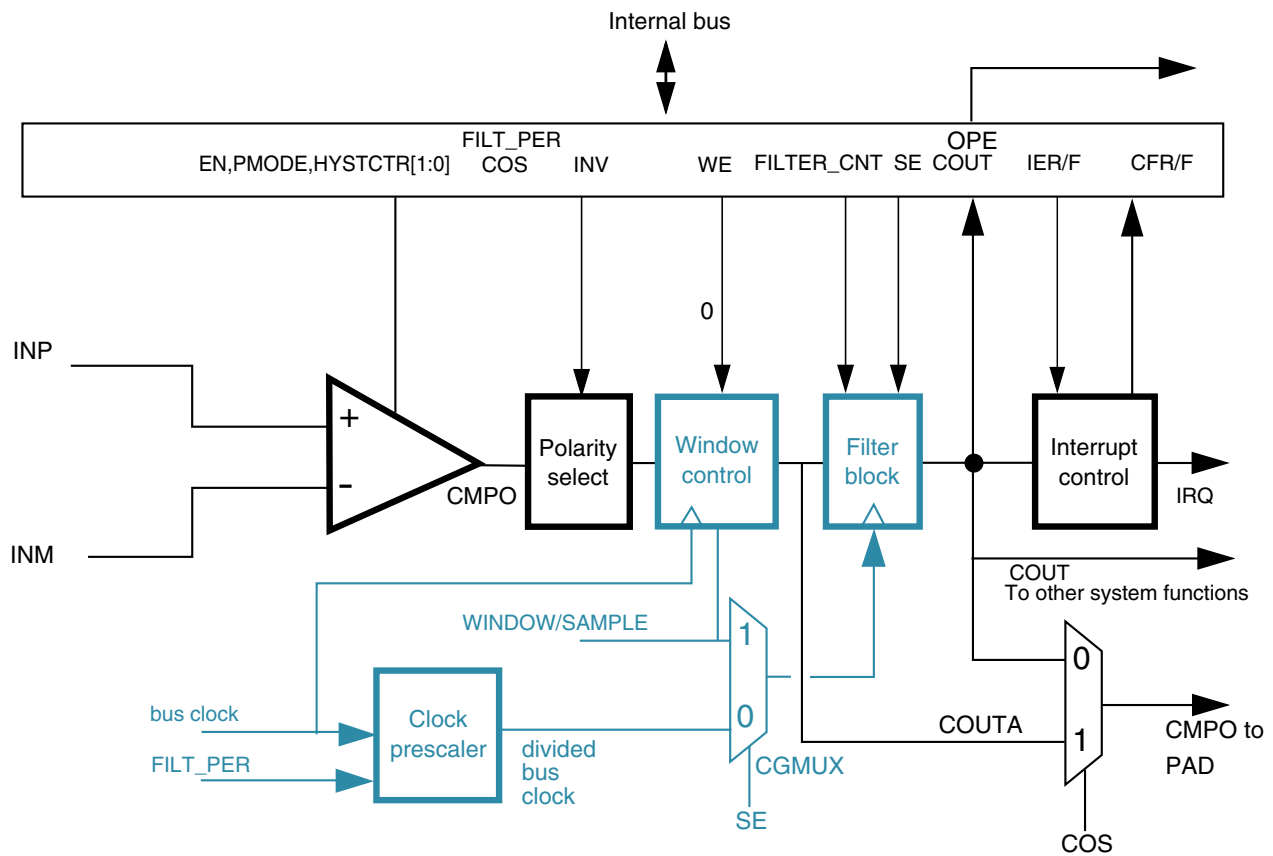


Figure 31-21. Comparator operation in Continuous mode

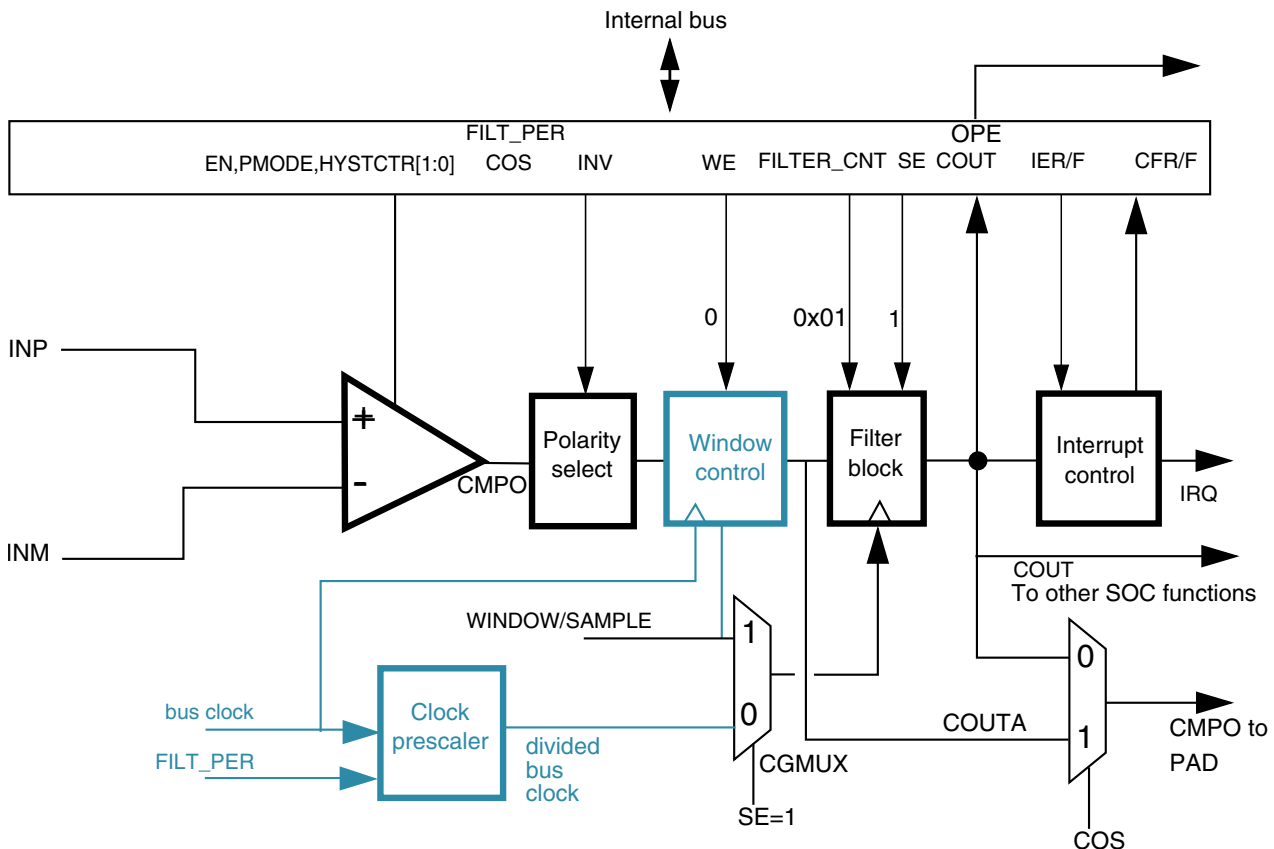


**NOTE**

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 unlocked mode. COUT and COUTA are identical.

For control configurations which result in disabling the filter block, see the [Filter Block Bypass Logic](#) diagram.

**31.3.1.3 Sampled, Non-Filtered mode (#s 3A & 3B)**

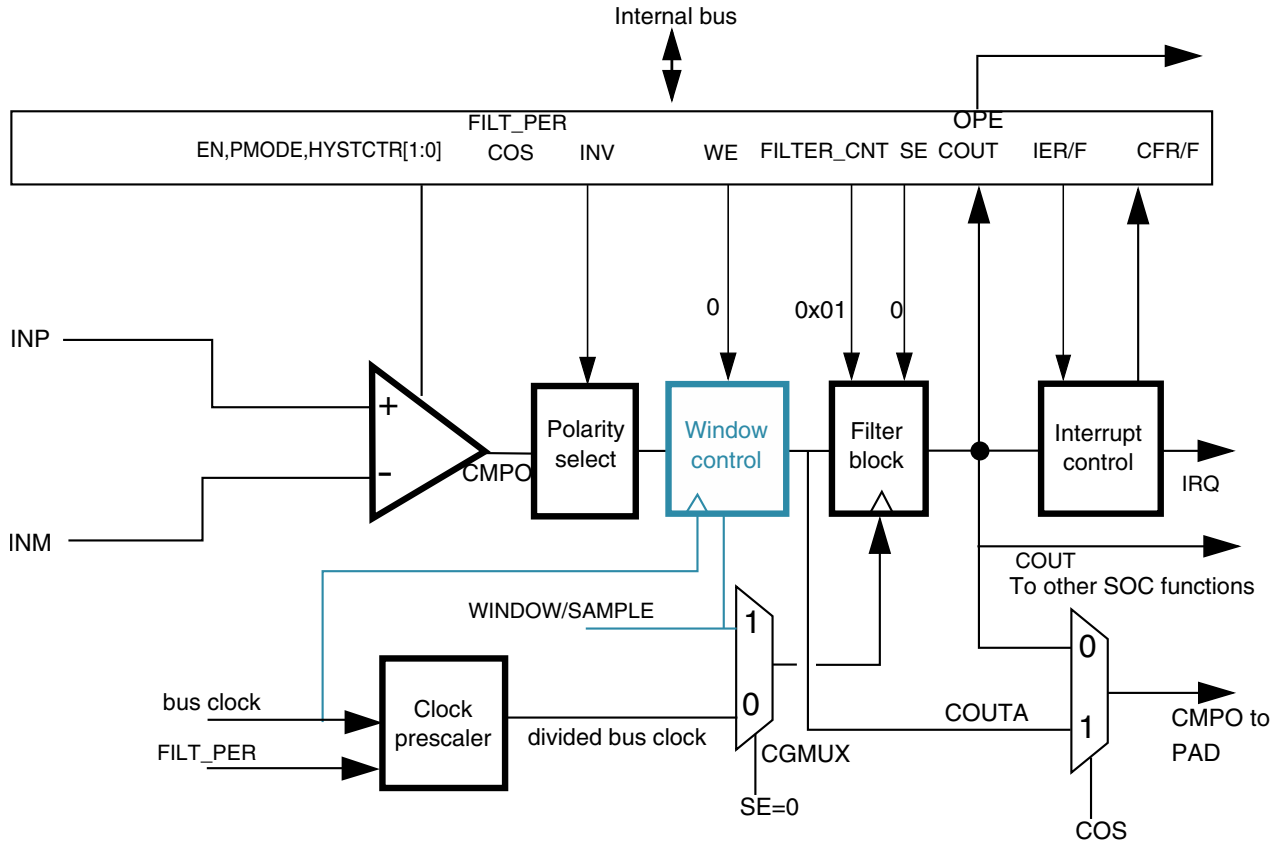
**Figure 31-22. Sampled, Non-Filtered (# 3A): sampling point externally driven**

In Sampled, Non-Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unlocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising-edge is detected on the filter block clock input.

## Functional description

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 31-23. Sampled, Non-Filtered (# 3B): sampling interval internally derived**

### 31.3.1.4 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 unlocked. 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.

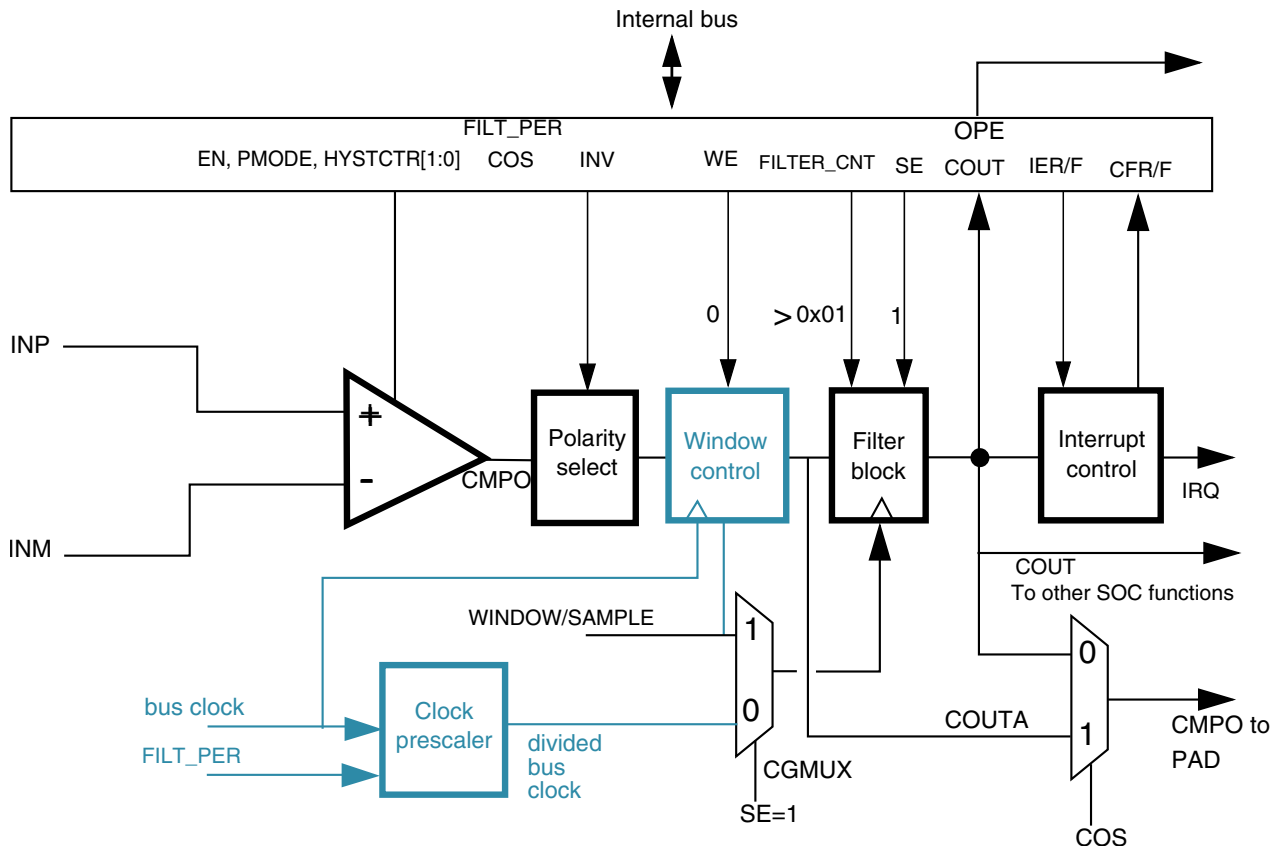
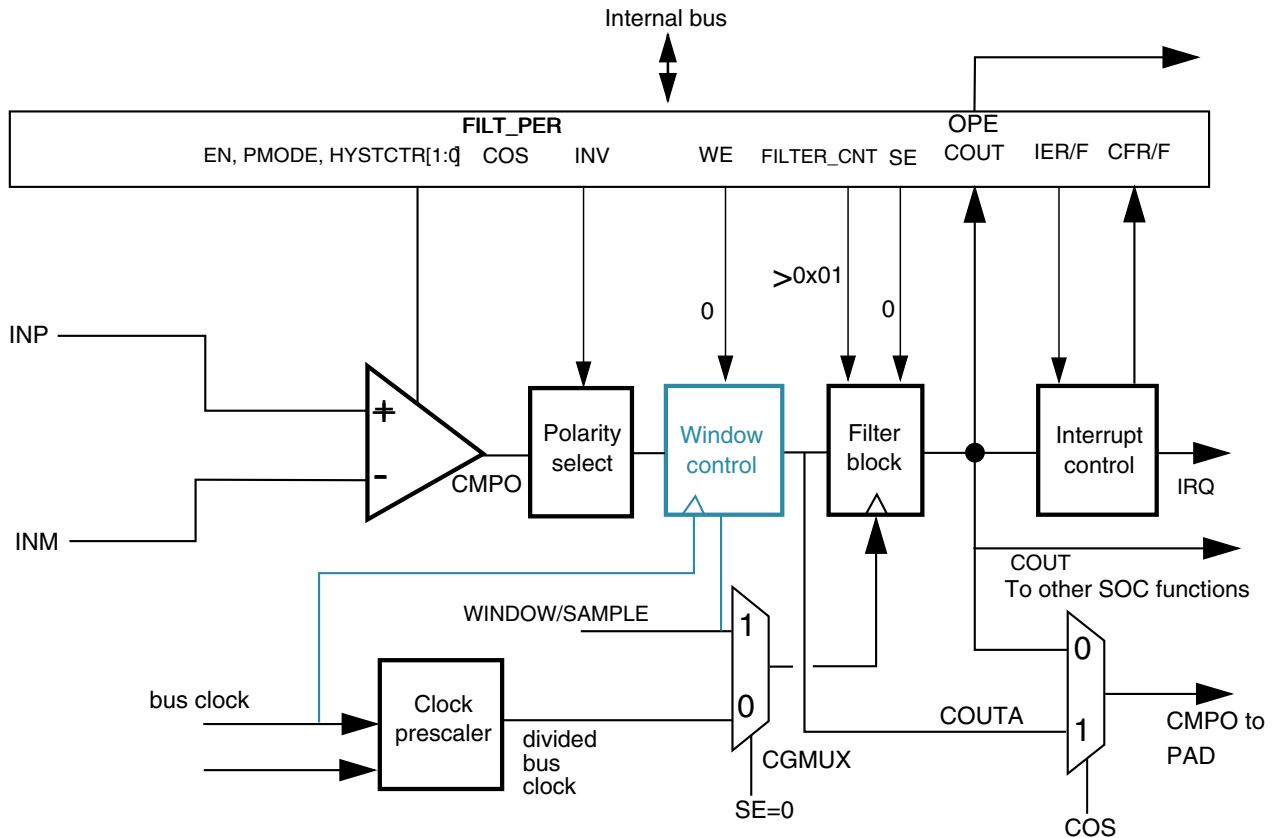


Figure 31-24. Sampled, Filtered (# 4A): sampling point externally driven



**Figure 31-25. 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.

### 31.3.1.5 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.

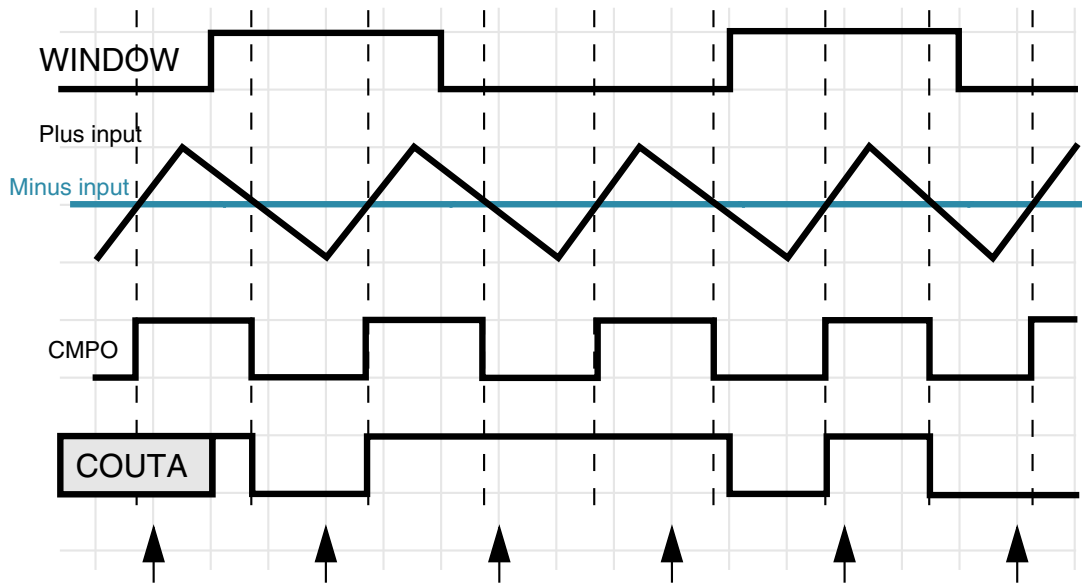


Figure 31-26. Windowed mode operation

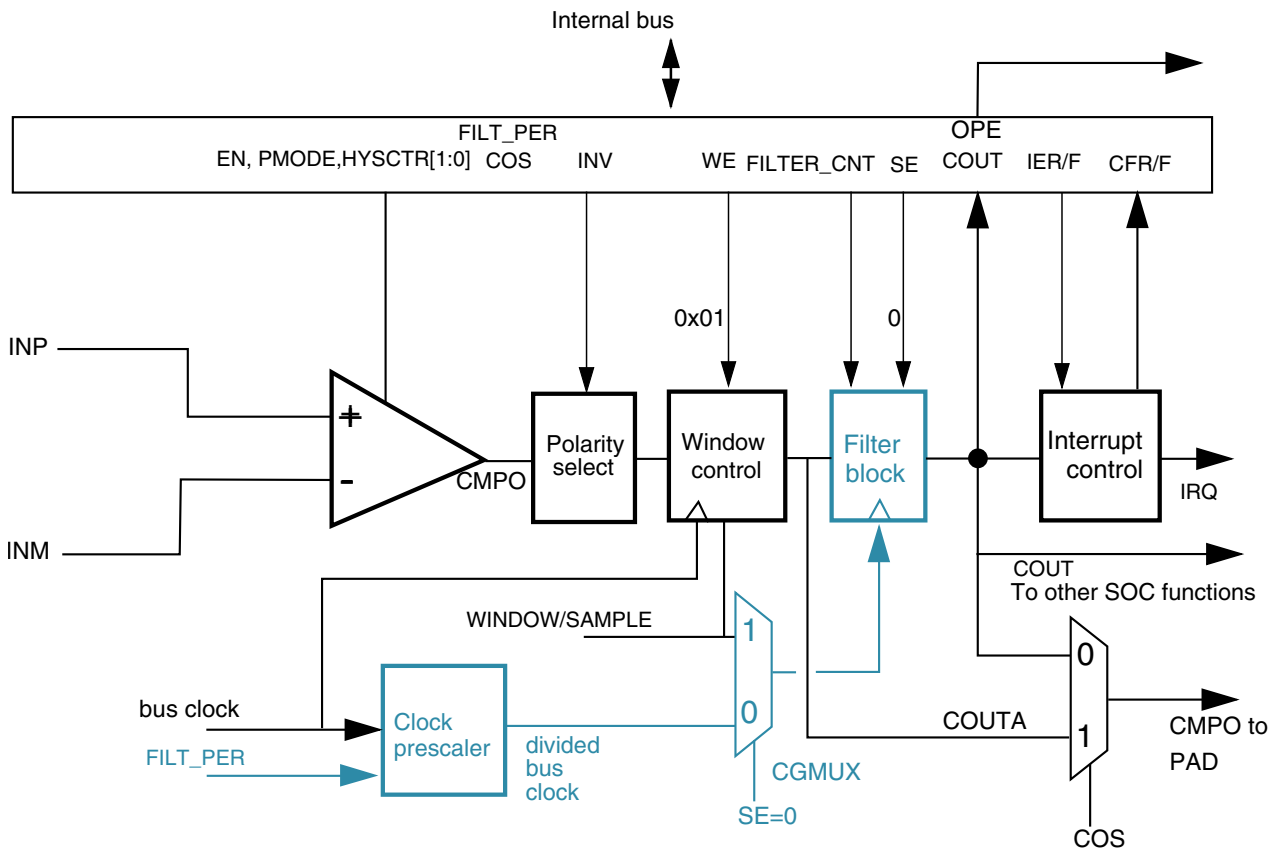


Figure 31-27. Windowed mode

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.

### 31.3.1.6 Windowed/Resampled mode (# 6)

The following figure uses the same input stimulus shown in Figure 31-26, 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.

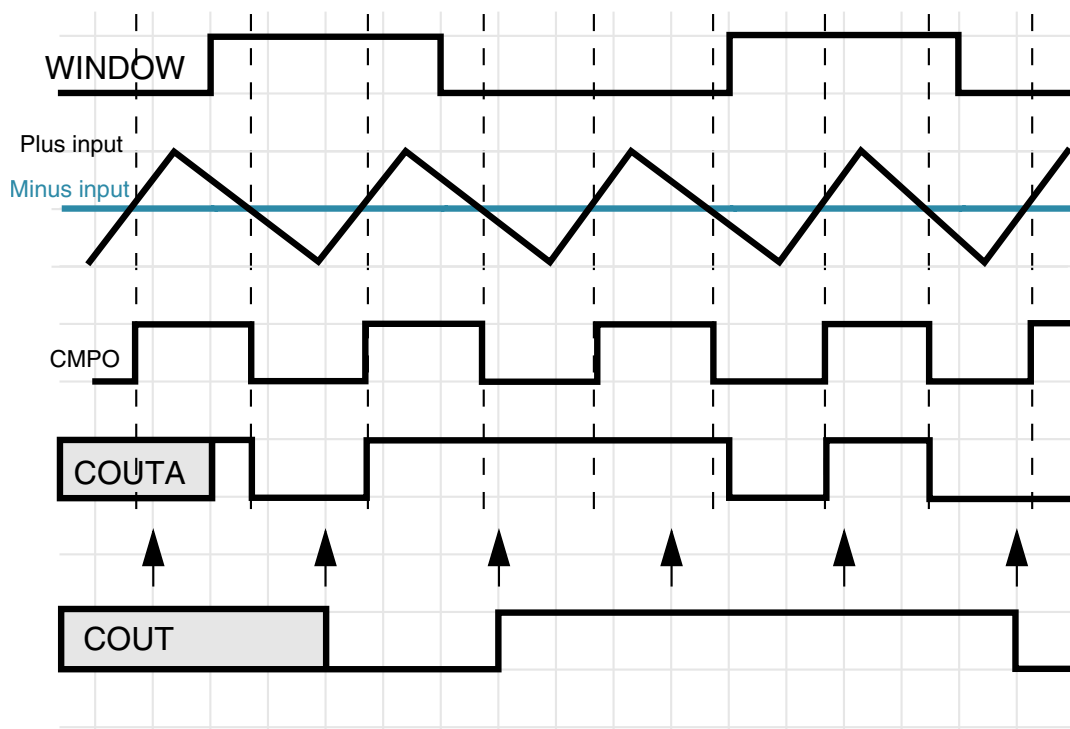


Figure 31-28. Windowed/resampled mode operation

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.

### 31.3.1.7 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 +  $((CR0[FILTER\_CNT] * FPR[FILT\_PER]) + 1) * \text{bus clock}$  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.

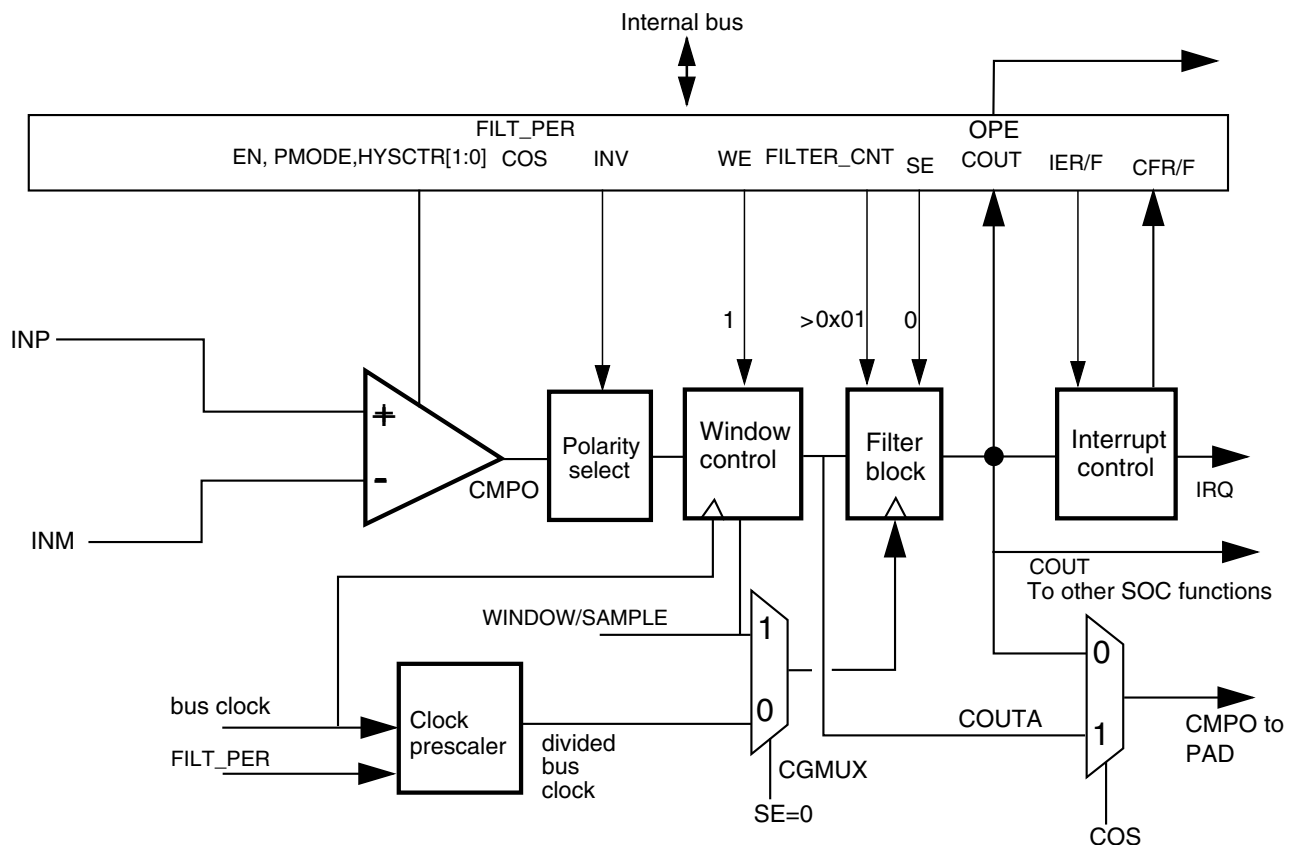


Figure 31-29. Windowed/Filtered mode

## 31.3.2 Power modes

### 31.3.2.1 Wait mode operation

During Wait and VLPW modes, the CMP, if enabled, continues to operate normally and a CMP interrupt can wake the MCU.

### 31.3.2.2 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.

### 31.3.2.3 Low-Leakage mode operation

When the chip is in Low-Leakage modes:

- The CMP module is partially functional and is limited to Low-Speed mode, regardless of CR1[PMODE] setting
- Windowed, Sampled, and Filtered modes are not supported
- The CMP output pin is latched and does not reflect the compare output state.

The positive- and negative-input voltage can be supplied from external pins or the DAC output. The MCU can be brought out of the Low-Leakage mode if a compare event occurs and the CMP interrupt is enabled. After wakeup from low-leakage modes, the CMP module is in the reset state except for SCR[CFF] and SCR[CFR].

### 31.3.3 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.

### 31.3.4 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.

#### 31.3.4.1 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 initialized, 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.

### 31.3.4.2 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 31-23. Comparator sample/filter maximum latencies**

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation	Maximum latency <sup>1</sup>
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		$T_{PD} + (CR0[FILTER\_CNT] * FPR[FILT\_PER] * T_{per}) + T_{per}$

Table continues on the next page...

**Table 31-23. Comparator sample/filter maximum latencies (continued)**

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation	Maximum latency <sup>1</sup>
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	$T_{PD} + (CR0[FILTER\_CNT] * FPR[FILT\_PER] * T_{per}) + 2T_{per}$

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.

## 31.4 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

## 31.5 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.

### 31.6 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.

## 31.7 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.

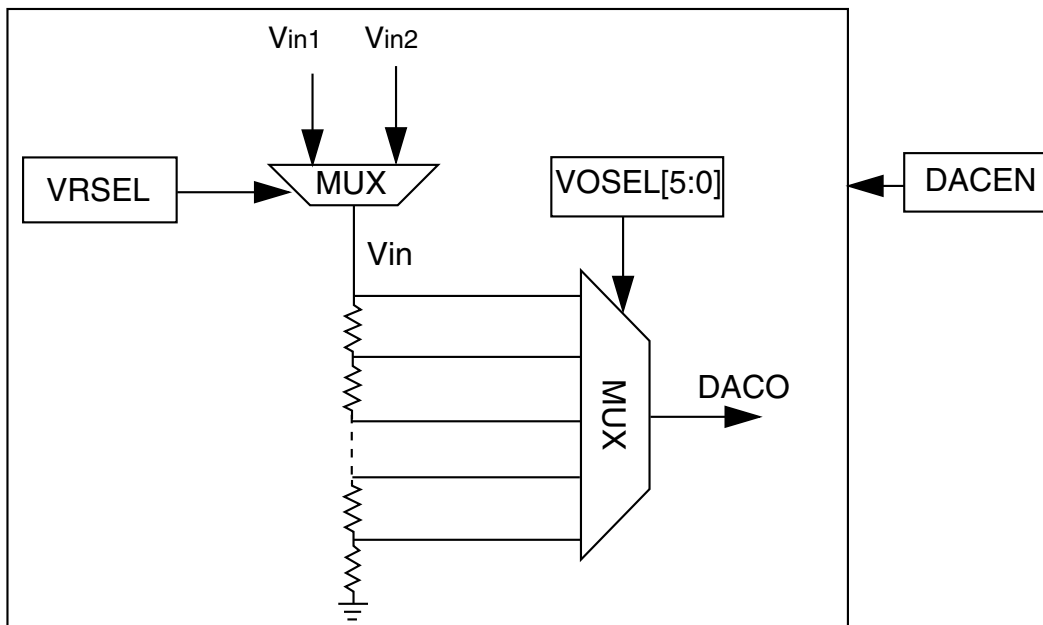


Figure 31-30. 6-bit DAC block diagram

## 31.8 DAC functional description

This section provides DAC functional description information.

### 31.8.1 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

## **31.9 DAC resets**

This module has a single reset input, corresponding to the chip-wide peripheral reset.

## **31.10 DAC clocks**

This module has a single clock input, the bus clock.

## **31.11 DAC interrupts**

This module has no interrupts.

# Chapter 32

## 12-bit Digital-to-Analog Converter (DAC)

### 32.1 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.

### 32.2 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
- 16-word data buffer supported with configurable watermark and multiple operation modes
- DMA support

### 32.3 Block diagram

The block diagram of the DAC module is as follows:

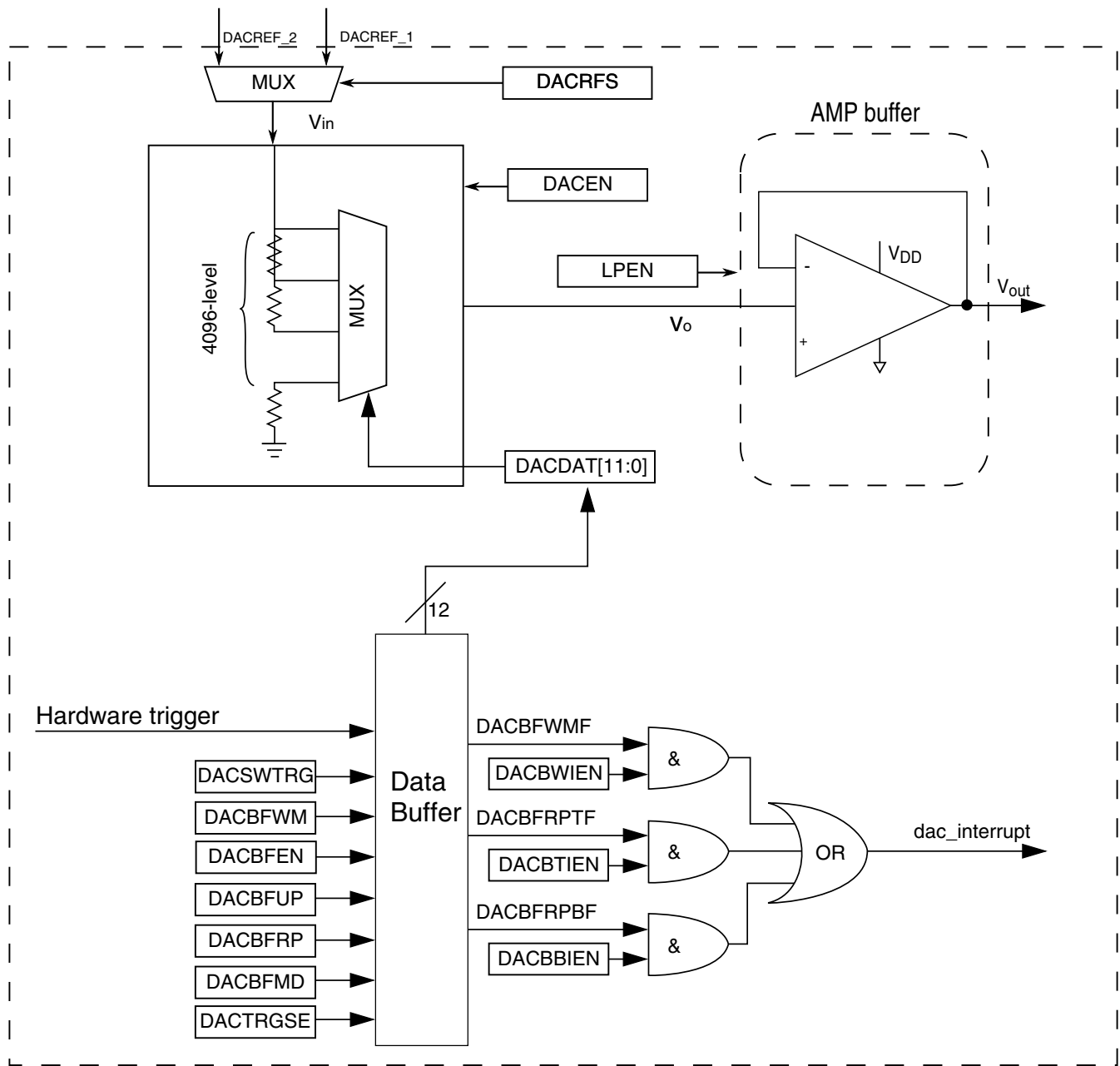


Figure 32-1. DAC block diagram

## 32.4 Memory map/register definition

The DAC has registers to control analog comparator and programmable voltage divider to perform the digital-to-analog functions.

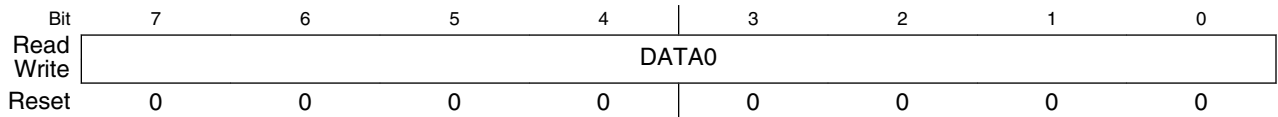


## DAC memory map

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	<a href="#">32.4.1/674</a>
4003_F001	DAC Data High Register (DAC0_DAT0H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F002	DAC Data Low Register (DAC0_DAT1L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F003	DAC Data High Register (DAC0_DAT1H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F004	DAC Data Low Register (DAC0_DAT2L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F005	DAC Data High Register (DAC0_DAT2H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F006	DAC Data Low Register (DAC0_DAT3L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F007	DAC Data High Register (DAC0_DAT3H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F008	DAC Data Low Register (DAC0_DAT4L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F009	DAC Data High Register (DAC0_DAT4H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F00A	DAC Data Low Register (DAC0_DAT5L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F00B	DAC Data High Register (DAC0_DAT5H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F00C	DAC Data Low Register (DAC0_DAT6L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F00D	DAC Data High Register (DAC0_DAT6H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F00E	DAC Data Low Register (DAC0_DAT7L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F00F	DAC Data High Register (DAC0_DAT7H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F010	DAC Data Low Register (DAC0_DAT8L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F011	DAC Data High Register (DAC0_DAT8H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F012	DAC Data Low Register (DAC0_DAT9L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F013	DAC Data High Register (DAC0_DAT9H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F014	DAC Data Low Register (DAC0_DAT10L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F015	DAC Data High Register (DAC0_DAT10H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F016	DAC Data Low Register (DAC0_DAT11L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F017	DAC Data High Register (DAC0_DAT11H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F018	DAC Data Low Register (DAC0_DAT12L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F019	DAC Data High Register (DAC0_DAT12H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F01A	DAC Data Low Register (DAC0_DAT13L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F01B	DAC Data High Register (DAC0_DAT13H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F01C	DAC Data Low Register (DAC0_DAT14L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F01D	DAC Data High Register (DAC0_DAT14H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F01E	DAC Data Low Register (DAC0_DAT15L)	8	R/W	00h	<a href="#">32.4.1/674</a>
4003_F01F	DAC Data High Register (DAC0_DAT15H)	8	R/W	00h	<a href="#">32.4.2/674</a>
4003_F020	DAC Status Register (DAC0_SR)	8	R/W	02h	<a href="#">32.4.3/675</a>
4003_F021	DAC Control Register (DAC0_C0)	8	R/W	00h	<a href="#">32.4.4/676</a>
4003_F022	DAC Control Register 1 (DAC0_C1)	8	R/W	00h	<a href="#">32.4.5/677</a>
4003_F023	DAC Control Register 2 (DAC0_C2)	8	R/W	0Fh	<a href="#">32.4.6/678</a>

### 32.4.1 DAC Data Low Register (DACx\_DATnL)

Address: 4003\_F000h base + 0h offset + (2d × i), where i=0d to 15d

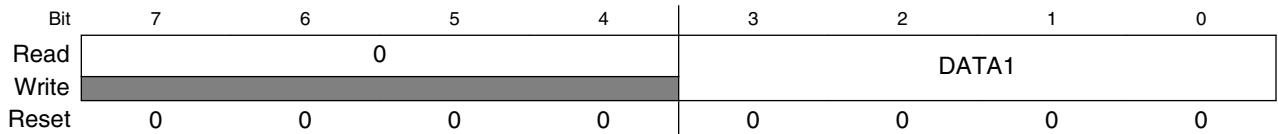


#### DACx\_DATnL field descriptions

Field	Description
7-0 DATA0	<p>DATA0</p> <p>When the DAC buffer is not enabled, DATA[11:0] controls the output voltage based on the following formula: <math>V_{out} = V_{in} * (1 + DACDAT0[11:0])/4096</math></p> <p>When the DAC buffer is enabled, DATA is mapped to the 16-word buffer.</p>

### 32.4.2 DAC Data High Register (DACx\_DATnH)

Address: 4003\_F000h base + 1h offset + (2d × i), where i=0d to 15d



#### DACx\_DATnH field descriptions

Field	Description
7-4 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
3-0 DATA1	<p>DATA1</p> <p>When the DAC Buffer is not enabled, DATA[11:0] controls the output voltage based on the following formula. <math>V_{out} = V_{in} * (1 + DACDAT0[11:0])/4096</math></p> <p>When the DAC buffer is enabled, DATA[11:0] is mapped to the 16-word buffer.</p>

### 32.4.3 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.

Address: 4003\_F000h base + 20h offset = 4003\_F020h

Bit	7	6	5	4	3	2	1	0
Read	0					DACBFWM	DACBFRPT	DACBFRPB
Write						F	F	F
Reset	0	0	0	0	0	0	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  This bit is set if the remaining FIFO data is less than the watermark setting. It is cleared automatically by writing data into FIFO by DMA or CPU. Write to this bit is ignored in FIFO mode.  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  In FIFO mode, it is FIFO nearly empty flag. It is set when only one data remains in FIFO. Any DAC trigger does not increase the Read Pointer if this bit is set to avoid any possible glitch or abrupt change at DAC output. It is cleared automatically if FIFO is not empty.  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  In FIFO mode, it is FIFO FULL status bit. It means FIFO read pointer equals Write Pointer because of Write Pointer increase. If this bit is set, any write to FIFO from either DMA or CPU is ignored by DAC. It is cleared if there is any DAC trigger making the DAC read pointer increase. Write to this bit is ignored in FIFO mode.  0 The DAC buffer read pointer is not equal to C2[DACBFUP]. 1 The DAC buffer read pointer is equal to C2[DACBFUP].

### 32.4.4 DAC Control Register (DACx\_C0)

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			L	DACSWTRG				
Reset	0	0	0	0	0	0	0	0

#### DACx\_C0 field descriptions

Field	Description
7 DACEN	<p>DAC Enable</p> <p>Starts the Programmable Reference Generator operation.</p> <p>0 The DAC system is disabled. 1 The DAC system is enabled.</p>
6 DACRFS	<p>DAC Reference Select</p> <p>0 The DAC selects DACREF_1 as the reference voltage. 1 The DAC selects DACREF_2 as the reference voltage.</p>
5 DACTRGSEL	<p>DAC Trigger Select</p> <p>0 The DAC hardware trigger is selected. 1 The DAC software trigger is selected.</p>
4 DACSCTR	<p>DAC Software Trigger</p> <p>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.</p> <p>0 The DAC soft trigger is not valid. 1 The DAC soft trigger is valid.</p>
3 LPEN	<p>DAC Low Power Control</p> <p><b>NOTE:</b> See the 12-bit DAC electrical characteristics of the device data sheet for details on the impact of the modes below.</p> <p>0 High-Power mode 1 Low-Power mode</p>
2 DACBWIEN	<p>DAC Buffer Watermark Interrupt Enable</p> <p>0 The DAC buffer watermark interrupt is disabled. 1 The DAC buffer watermark interrupt is enabled.</p>
1 DACBTIEN	<p>DAC Buffer Read Pointer Top Flag Interrupt Enable</p> <p>0 The DAC buffer read pointer top flag interrupt is disabled. 1 The DAC buffer read pointer top flag interrupt is enabled.</p>
0 DACBBIEN	<p>DAC Buffer Read Pointer Bottom Flag Interrupt Enable</p>

Table continues on the next page...

## DACx\_C0 field descriptions (continued)

Field	Description
0	The DAC buffer read pointer bottom flag interrupt is disabled.
1	The DAC buffer read pointer bottom flag interrupt is enabled.

## 32.4.5 DAC Control Register 1 (DACx\_C1)

Address: 4003\_F000h base + 22h offset = 4003\_F022h

Bit	7	6	5	4	3	2	1	0	
Read	DMAEN	0			DACBFWM		DACBFMD		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.
6–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–3 DACBFWM	DAC Buffer Watermark Select In normal mode it controls when SR[DACBFWMF] is set. When the DAC buffer read pointer reaches the word defined by this field, which is 1–4 words away from the upper limit (DACBUP), SR[DACBFWMF] will be set. This allows user configuration of the watermark interrupt. In FIFO mode, it is FIFO watermark select field. 00 In normal mode, 1 word . In FIFO mode, 2 or less than 2 data remaining in FIFO will set watermark status bit. 01 In normal mode, 2 words . In FIFO mode, Max/4 or less than Max/4 data remaining in FIFO will set watermark status bit. 10 In normal mode, 3 words . In FIFO mode, Max/2 or less than Max/2 data remaining in FIFO will set watermark status bit. 11 In normal mode, 4 words . In FIFO mode, Max-2 or less than Max-2 data remaining in FIFO will set watermark status bit.
2–1 DACBFMD	DAC Buffer Work Mode Select 00 Normal mode 01 Swing mode 10 One-Time Scan mode 11 FIFO mode
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.

### 32.4.6 DAC Control Register 2 (DACx\_C2)

Address: 4003\_F000h base + 23h offset = 4003\_F023h

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

#### DACx\_C2 field descriptions

Field	Description
7–4 DACBFRP	DAC Buffer Read Pointer  In normal mode it keeps the current value of the buffer read pointer. FIFO mode, it is the FIFO read pointer. It is writable in FIFO mode. User can configure it to same address to reset FIFO as empty.
3–0 DACBFUP	DAC Buffer Upper Limit  In normal mode it selects the upper limit of the DAC buffer. The buffer read pointer cannot exceed it. In FIFO mode it is the FIFO write pointer. User cannot set Buffer Up limit in FIFO mode. In Normal mode its reset value is MAX. When IP is configured to FIFO mode, this register becomes Write_Pointer, and its value is initially set to equal READ_POINTER automatically, and the FIFO status is empty. It is writable and user can configure it to the same address to reset FIFO as empty.

## 32.5 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 module introduction for information on the source 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  $V_{in}/4096$ .

### 32.5.1 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 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 any hardware or software trigger event occurs.

The data buffer can be configured to operate in Normal mode, Swing mode, One-Time Scan mode or FIFO 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].

### 32.5.1.1 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].

### 32.5.1.2 Modes of DAC data buffer operation

The following table describes the different modes of data buffer operation for the DAC module.

**Table 32-79. 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 Swing mode	This mode is similar to the normal mode. However, when the read pointer reaches the upper limit, it does not go to 0. It will descend by 1 in the next trigger events until 0 is reached.
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.  <b>NOTE:</b> If the software set the read pointer to the upper limit, the read pointer will not advance in this mode.

*Table continues on the next page...*

Table 32-79. Modes of DAC data buffer operation (continued)

Modes	Description
FIFO Mode	<p>In FIFO mode, the buffer is organized as a FIFO. For a valid write to any DACDATx, the data is put into the FIFO, and the write pointer is automatically incremented. The module is connected internally to a 32bit interface. For any 16bit or 8bit FIFO access, address bit[1] needs to be 0; otherwise, the write is ignored. For any 32bit FIFO access, the Write_Pointer needs to be an EVEN number; otherwise, the write is ignored.</p> <p><b>NOTE:</b> A successful 32bit FIFO write will increase the write pointer by 2. Any write will cause the FIFO over-flow will be ignored, the cases includes: 1.FIFO is full, the write will be ignored. 2.FIFO is nearly full (FIFO_SIZE-1), 32bit write will be ignored.</p> <p><b>NOTE:</b> For 8bit write, address bit[0] determine which byte lane will be written to the FIFO according to little endian alignment. Only both byte lanes are written will the write pointer increase. User need to make sure 8bit access happened in pair and both upper &amp; lower bytes are written. There is no requirement on which byte write first. In FIFO mode, there is no change to read access of DACDATx (from normal mode), read to DACDATx will return the DATA addressed by the access address to the data buffer, and both write pointer and read pointer in FIFO mode will NOT be changed by read access. FIFO write can be happened when DAC is not enabled for 1st data conversion enable. But FIFO mode need to work at buffer Enabled at DACC1[DACBFEN].</p> <p>In FIFO mode, the DATA BUF will be organized as FIFO.</p>

### 32.5.2 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.

### 32.5.3 Resets

During reset, the DAC is configured in the default mode and is disabled.



### 32.5.4 Low-Power mode operation

The following table shows the wait mode and the stop mode operation of the DAC module.

**Table 32-80. Modes of operation**

Modes of operation	Description
Wait mode	The DAC will operate normally, if enabled.
Stop mode	<p>If enabled, the DAC module continues to operate in Normal Stop mode and the output voltage will hold the value before stop.</p> <p>In low-power stop modes, the DAC is fully shut down.</p>

#### 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 33

# Voltage Reference (VREFV1)

### 33.1 Introduction

#### NOTE

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

The Voltage Reference(VREF) is intended to supply an accurate voltage output that can be trimmed in 0.5 mV steps. The VREF can be used in applications to provide a reference voltage to external devices or used internally as a reference to analog peripherals such as the ADC, DAC, or CMP. The voltage reference has three operating modes that provide different levels of supply rejection and power consumption..

The following figure is a block diagram of the Voltage Reference.

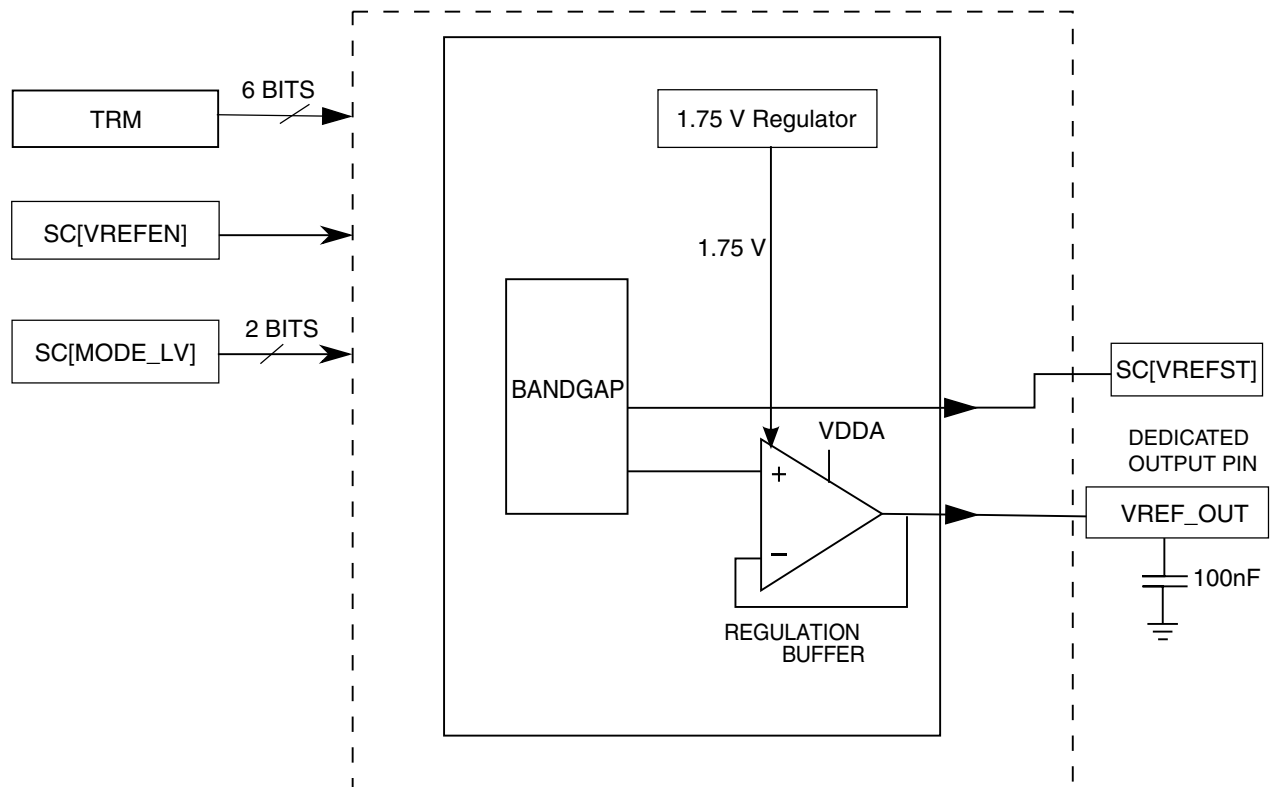


Figure 33-1. Voltage reference block diagram

### 33.1.1 Overview

The Voltage Reference provides a buffered reference voltage for use as an external reference. In addition, the buffered reference is available internally for use with on chip peripherals such as ADCs and DACs. Refer to the chip configuration details for a description of these options. The reference voltage is output on a dedicated output pin when the VREF is enabled. The Voltage Reference output can be trimmed with a resolution of 0.5mV by means of the TRM register TRIM[5:0] bitfield.

### 33.1.2 Features

The Voltage Reference has the following features:

- Programmable trim register with 0.5 mV steps, automatically loaded with factory trimmed value upon reset
- Programmable buffer mode selection:
  - Off

- Bandgap enabled/standby (output buffer disabled)
- Low power buffer mode (output buffer enabled)
- High power buffer mode (output buffer enabled)
- 1.2 V output at room temperature
- Dedicated output pin, VREF\_OUT

### 33.1.3 Modes of Operation

The Voltage Reference continues normal operation in Run, Wait, and Stop modes. The Voltage Reference can also run in Very Low Power Run (VLPR), Very Low Power Wait (VLPW) and Very Low Power Stop (VLPS). If it is desired to use the VREF regulator and/or the chop oscillator in the very low power modes, the system reference voltage (also referred to as the bandgap voltage reference) must be enabled in these modes. Refer to the chip configuration details for information on enabling this mode of operation. Having the VREF regulator enabled does increase current consumption. In very low power modes it may be desirable to disable the VREF regulator to minimize current consumption. Note however that the accuracy of the output voltage will be reduced (by as much as several mVs) when the VREF regulator is not used.

#### 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.

### 33.1.4 VREF Signal Descriptions

The following table shows the Voltage Reference signals properties.

**Table 33-1. VREF Signal Descriptions**

Signal	Description	I/O
VREF_OUT	Internally-generated Voltage Reference output	O

#### NOTE

When the VREF output buffer is disabled, the status of the VREF\_OUT signal is high-impedance.

## 33.2 Memory Map and Register Definition

### VREF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4007_4000	VREF Trim Register (VREF_TRM)	8	R/W	See section	33.2.1/686
4007_4001	VREF Status and Control Register (VREF_SC)	8	R/W	00h	33.2.2/687

### 33.2.1 VREF Trim Register (VREF\_TRM)

This register contains bits that contain the trim data for the Voltage Reference.

Address: 4007\_4000h base + 0h offset = 4007\_4000h

Bit	7	6	5	4	3	2	1	0
Read	Reserved	CHOPEN	TRIM					
Write								
Reset	x*	0	x*	x*	x*	x*	x*	x*

\* Notes:

- x = Undefined at reset.

### VREF\_TRM field descriptions

Field	Description
7 Reserved	This field is reserved. Upon reset this value is loaded with a factory trim value.
6 CHOPEN	Chop oscillator enable. When set, internal chopping operation is enabled and the internal analog offset will be minimized.  This bit is set during factory trimming of the VREF voltage. This bit should be written to 1 to achieve the performance stated in the data sheet.  If the internal voltage regulator is being used (REGEN bit is set to 1), the chop oscillator must also be enabled.  If the chop oscillator is to be used in very low power modes, the system (bandgap) voltage reference must also be enabled. See the chip-specific VREF information (also known as "chip configuration" details) for a description of how this can be achieved.  0 Chop oscillator is disabled. 1 Chop oscillator is enabled.
5–0 TRIM	Trim bits  These bits change the resulting VREF by approximately $\pm 0.5$ mV for each step.  <b>NOTE:</b> Min = minimum and max = maximum voltage reference output. For minimum and maximum voltage reference output values, refer to the Data Sheet for this chip.

Table continues on the next page...

## VREF\_TRM field descriptions (continued)

Field	Description
000000	Min
....	....
111111	Max

## 33.2.2 VREF Status and Control Register (VREF\_SC)

This register contains the control bits used to enable the internal voltage reference and to select the buffer mode to be used.

Address: 4007\_4000h base + 1h offset = 4007\_4001h

Bit	7	6	5	4	3	2	1	0
Read	VREFEN	REGEN	ICOMPEN	0	0	VREFST	MODE_LV	
Write								
Reset	0	0	0	0	0	0	0	0

## VREF\_SC field descriptions

Field	Description
7 VREFEN	<p>Internal Voltage Reference enable</p> <p>This bit is used to enable the bandgap reference within the Voltage Reference module.</p> <p><b>NOTE:</b> After the VREF is enabled, turning off the clock to the VREF module via the corresponding clock gate register will not disable the VREF. VREF must be disabled via this VREFEN bit.</p> <p>0 The module is disabled. 1 The module is enabled.</p>
6 REGEN	<p>Regulator enable</p> <p>This bit is used to enable the internal 1.75 V regulator to produce a constant internal voltage supply in order to reduce the sensitivity to external supply noise and variation. If it is desired to keep the regulator enabled in very low power modes, refer to the Chip Configuration details for a description on how this can be achieved.</p> <p>This bit should be written to 1 to achieve the performance stated in the data sheet.</p> <p><b>NOTE:</b> See section "Internal voltage regulator" for details on the required sequence to enable the internal regulator.</p> <p>0 Internal 1.75 V regulator is disabled. 1 Internal 1.75 V regulator is enabled.</p>
5 ICOMPEN	<p>Second order curvature compensation enable</p> <p>This bit should be written to 1 to achieve the performance stated in the data sheet.</p>

Table continues on the next page...

## VREF\_SC field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
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 VREFST	Internal Voltage Reference stable  This bit indicates that the bandgap reference within the Voltage Reference module has completed its startup and stabilization.  <b>NOTE:</b> This bit is valid only when the chop oscillator is not being used.  0 The module is disabled or not stable. 1 The module is stable.
1–0 MODE_LV	Buffer Mode selection  These bits select the buffer modes for the Voltage Reference module.  00 Bandgap on only, for stabilization and startup 01 High power buffer mode enabled 10 Low-power buffer mode enabled 11 Reserved

### 33.3 Functional Description

The Voltage Reference is a bandgap buffer system. Unity gain amplifiers are used.

The VREF\_OUT signal can be used by both internal and external peripherals in low and high power buffer mode. A 100 nF capacitor must always be connected between VREF\_OUT and VSSA if the VREF is being used.

The following table shows all possible function configurations of the Voltage Reference.

**Table 33-5. Voltage Reference function configurations**

SC[VREFEN]	SC[MODE_LV]	Configuration	Functionality
0	X	Voltage Reference disabled	Off
1	00	Voltage Reference enabled, bandgap on only	Startup and standby
1	01	Voltage Reference enabled, high-power buffer on	VREF_OUT available for internal and external use. 100 nF capacitor is required.
1	10	Voltage Reference enabled, low power buffer on	VREF_OUT available for internal and external use. 100 nF capacitor is required.
1	11	Reserved	Reserved



### 33.3.1 Voltage Reference Disabled, SC[VREFEN] = 0

When SC[VREFEN] = 0, the Voltage Reference is disabled, the VREF bandgap and the output buffers are disabled. The Voltage Reference is in off mode.

### 33.3.2 Voltage Reference Enabled, SC[VREFEN] = 1

When SC[VREFEN] = 1, the Voltage Reference is enabled, and different modes should be set by the SC[MODE\_LV] bits.

#### 33.3.2.1 SC[MODE\_LV]=00

The internal VREF bandgap is enabled to generate an accurate 1.2 V output that can be trimmed with the TRM register's TRIM[5:0] bitfield. The bandgap requires some time for startup and stabilization. SC[VREFST] can be monitored to determine if the stabilization and startup is complete when the chop oscillator is not enabled.

If the chop oscillator is being used, the internal bandgap reference voltage settles within the chop oscillator start up time, T<sub>chop\_osc\_stup</sub>.

The output buffer is disabled in this mode, and there is no buffered voltage output. The Voltage Reference is in standby mode. If this mode is first selected and the low power or high power buffer mode is subsequently enabled, there will be a delay before the buffer output is settled at the final value. This is the buffer start up delay (T<sub>stup</sub>) and the value is specified in the appropriate device data sheet.

#### 33.3.2.2 SC[MODE\_LV] = 01

The internal VREF bandgap is on. The high power buffer is enabled to generate a buffered 1.2 V voltage to VREF\_OUT. It can also be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

If this mode is entered from the standby mode (SC[MODE\_LV] = 00, SC[VREFEN] = 1) there will be a delay before the buffer output is settled at the final value. This is the buffer start up delay (T<sub>stup</sub>) and the value is specified in the appropriate device data sheet. If this mode is entered when the VREF module is enabled then you must wait the longer of

Tstup or until SC[VREFST] = 1 when the chop oscillator is not enabled. If the chop oscillator is being used, you must wait the time specified by Tchop\_osc\_stup (chop oscillator start up time) to ensure the VREF output has stabilized.

In this mode, a 100 nF capacitor is required to connect between the VREF\_OUT pin and VSSA.

### **33.3.2.3 SC[MODE\_LV] = 10**

The internal VREF bandgap is on. The low power buffer is enabled to generate a buffered 1.2 V voltage to VREF\_OUT. It can also be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

If this mode is entered from the standby mode (SC[MODE\_LV] = 00, SC[VREFEN] = 1) there will be a delay before the buffer output is settled at the final value. This is the buffer start up delay (Tstup) and the value is specified in the appropriate device data sheet. If this mode is entered when the VREF module is enabled then you must wait the longer of Tstup or until SC[VREFST] = 1 when the chop oscillator is not enabled. If the chop oscillator is being used, you must wait the time specified by Tchop\_osc\_stup (chop oscillator start up time) to ensure the VREF output has stabilized.

In this mode, a 100 nF capacitor is required to connect between the VREF\_OUT pin and VSSA.

### **33.3.2.4 SC[MODE\_LV] = 11**

Reserved

## **33.3.3 Internal voltage regulator**

The VREF module contains an internal voltage regulator that can be enabled to provide additional supply noise rejection. It is recommended that when possible, this regulator be enabled to provide the optimum VREF performance.

If the internal voltage regulator is being used, the chop oscillator must also be enabled. A specific sequence must be followed when enabling the internal regulator as follows:

1. Enable the chop oscillator (VREF\_TRM[CHOPEN] = 1)
2. Configure the VREF\_SC register to the desired settings with the internal regulator disabled, VREF\_SC[REGEN] = 0
3. Wait > 300ns

4. Enable the internal regulator by setting VREF\_SC[REGEN] to 1

### 33.4 Initialization/Application Information

The Voltage Reference requires some time for startup and stabilization. After SC[VREFEN] = 1, SC[VREFST] can be monitored to determine if the stabilization and startup is completed when the chop oscillator is not enabled. When the chop oscillator is enabled, the settling time of the internal bandgap reference is defined by Tchop\_osc\_stup (chop oscillator start up time). You must wait this time (Tchop\_osc\_stup) after the internal bandgap has been enabled to ensure the VREF internal reference voltage has stabilized.

When the Voltage Reference is already enabled and stabilized, changing SC[MODE\_LV] will not clear SC[VREFST] but there will be some startup time before the output voltage at the VREF\_OUT pin has settled. This is the buffer start up delay (Tstup) and the value is specified in the appropriate device data sheet. Also, there will be some settling time when a step change of the load current is applied to the VREF\_OUT pin. When the 1.75V VREF regulator is disabled, the VREF\_OUT voltage will be more sensitive to supply voltage variation. It is recommended to use this regulator to achieve optimum VREF\_OUT performance.

The TRM[CHOPEN], SC[REGEN] and SC[ICOMPEN] bits must be written to 1 to achieve the performance stated in the device data sheet.

#### NOTE

See section "Internal voltage regulator" for details on the required sequence to enable the internal regulator.



# Chapter 34

## Programmable Delay Block (PDB)

### 34.1 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.

#### 34.1.1 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 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.

## 34.1.2 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.

### **34.1.3 Back-to-back acknowledgment connections**

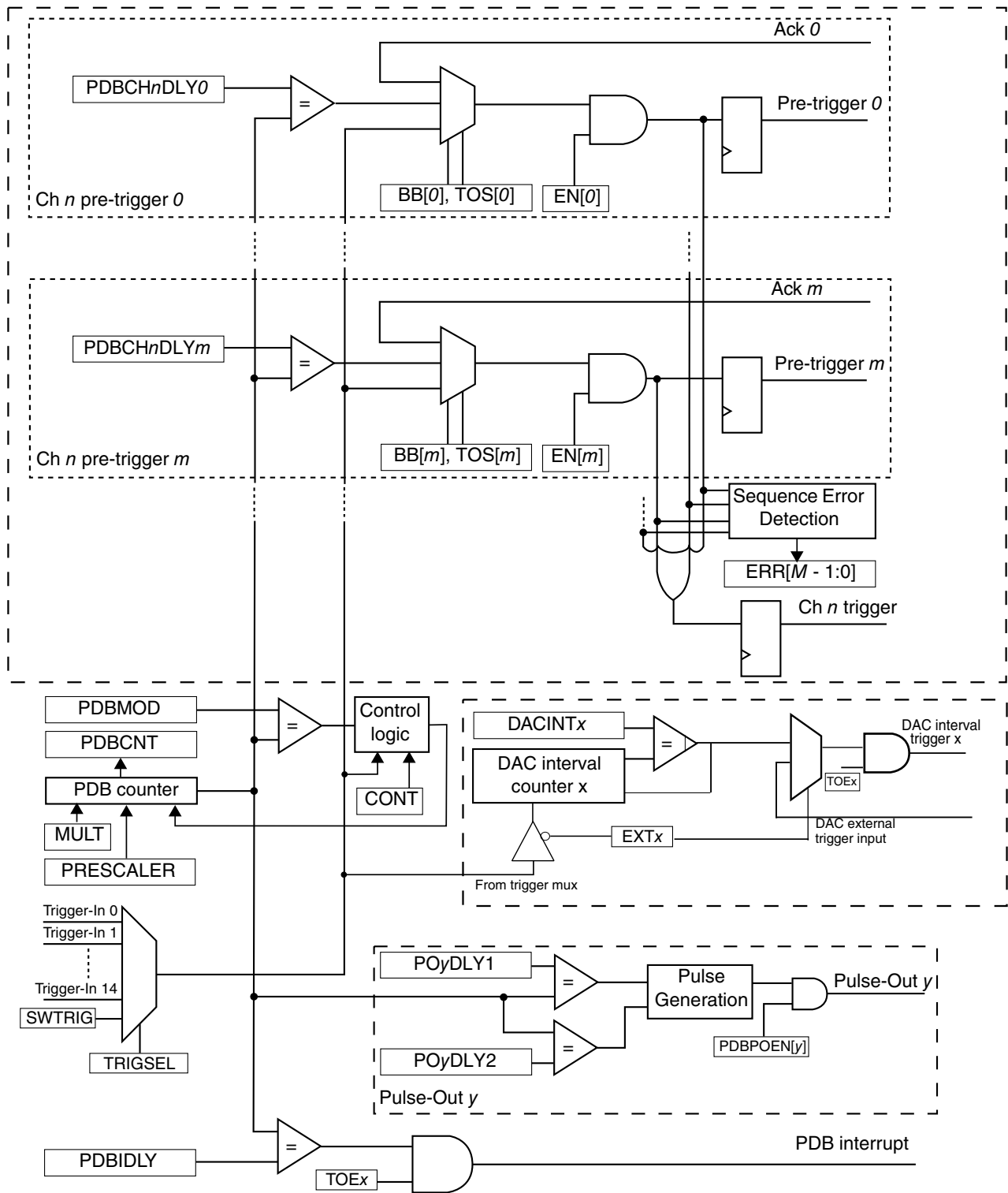
PDB back-to-back operation acknowledgment connections are chip-specific. For implementation, see the chip configuration information.

### **34.1.4 DAC External Trigger Input Connections**

The implementation of DAC external trigger inputs is chip-specific. See the chip configuration information for details.

### **34.1.5 Block diagram**

This diagram illustrates the major components of the PDB.



**Figure 34-1. 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.



### 34.1.6 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.

## 34.2 PDB signal descriptions

This table shows the detailed description of the external signal.

**Table 34-1. 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

## 34.3 Memory map and register definition

## PDB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_6000	Status and Control register (PDB0_SC)	32	R/W	0000_0000h	<a href="#">34.3.1/699</a>
4003_6004	Modulus register (PDB0_MOD)	32	R/W	0000_FFFFh	<a href="#">34.3.2/701</a>
4003_6008	Counter register (PDB0_CNT)	32	R	0000_0000h	<a href="#">34.3.3/702</a>
4003_600C	Interrupt Delay register (PDB0_IDLY)	32	R/W	0000_FFFFh	<a href="#">34.3.4/702</a>
4003_6010	Channel n Control register 1 (PDB0_CH0C1)	32	R/W	0000_0000h	<a href="#">34.3.5/703</a>
4003_6014	Channel n Status register (PDB0_CH0S)	32	R/W	0000_0000h	<a href="#">34.3.6/704</a>
4003_6018	Channel n Delay 0 register (PDB0_CH0DLY0)	32	R/W	0000_0000h	<a href="#">34.3.7/704</a>
4003_601C	Channel n Delay 1 register (PDB0_CH0DLY1)	32	R/W	0000_0000h	<a href="#">34.3.8/705</a>
4003_6038	Channel n Control register 1 (PDB0_CH1C1)	32	R/W	0000_0000h	<a href="#">34.3.5/703</a>
4003_603C	Channel n Status register (PDB0_CH1S)	32	R/W	0000_0000h	<a href="#">34.3.6/704</a>
4003_6040	Channel n Delay 0 register (PDB0_CH1DLY0)	32	R/W	0000_0000h	<a href="#">34.3.7/704</a>
4003_6044	Channel n Delay 1 register (PDB0_CH1DLY1)	32	R/W	0000_0000h	<a href="#">34.3.8/705</a>
4003_6150	DAC Interval Trigger n Control register (PDB0_DACINTC0)	32	R/W	0000_0000h	<a href="#">34.3.9/705</a>
4003_6154	DAC Interval n register (PDB0_DACINT0)	32	R/W	0000_0000h	<a href="#">34.3.10/706</a>
4003_6190	Pulse-Out n Enable register (PDB0_POEN)	32	R/W	0000_0000h	<a href="#">34.3.11/706</a>
4003_6194	Pulse-Out n Delay register (PDB0_PO0DLY)	32	R/W	0000_0000h	<a href="#">34.3.12/707</a>
4003_6198	Pulse-Out n Delay register (PDB0_PO1DLY)	32	R/W	0000_0000h	<a href="#">34.3.12/707</a>

### 34.3.1 Status and Control register (PDBx\_SC)

Address: 4003\_6000h base + 0h offset = 4003\_6000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0												LDMOD		PDBEIE	0	
W	[Reserved]												LDMOD		PDBEIE	SWTRIG	
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	DMAEN	PRESCALER				TRGSEL				PDBEN	PDBIF	PDBIE	0		MULT	CONT	LDOK
W	DMAEN	PRESCALER				TRGSEL				PDBEN	PDBIF	PDBIE	[Reserved]		MULT	CONT	LDOK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

#### 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 <sub>n</sub> DLY <sub>m</sub> , INT <sub>x</sub> , and POyDLY 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 reaches the MOD register value after 1 is written to LDOK. 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 reaches the MOD register value 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 this field is set, any of the PDB channel sequence error flags generates a PDB sequence error interrupt.

Table continues on the next page...

## PDBx\_SC field descriptions (continued)

Field	Description
	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 this field resets and restarts the counter. Writing 0 to this field has no effect. Reading this field results 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  000 Counting uses the peripheral clock divided by multiplication factor selected by MULT. 001 Counting uses the peripheral clock divided by twice of the multiplication factor selected by MULT. 010 Counting uses the peripheral clock divided by four times of the multiplication factor selected by MULT. 011 Counting uses the peripheral clock divided by eight times of the multiplication factor selected by MULT. 100 Counting uses the peripheral clock divided by 16 times of the multiplication factor selected by MULT. 101 Counting uses the peripheral clock divided by 32 times of the multiplication factor selected by MULT. 110 Counting uses the peripheral clock divided by 64 times of the multiplication factor selected by MULT. 111 Counting uses the peripheral clock divided by 128 times of the multiplication factor selected by MULT.
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. 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.

*Table continues on the next page...*

## PDBx\_SC field descriptions (continued)

Field	Description
7 PDBEN	PDB Enable 0 PDB disabled. Counter is off. 1 PDB enabled.
6 PDBIF	PDB Interrupt Flag This field is set when the counter value is equal to the IDLY register. Writing zero clears this field.
5 PDBIE	PDB Interrupt Enable Enables the PDB interrupt. When this field 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 this bit updates the internal registers of MOD, IDLY, CHnDLYm, DACINTx, and POyDLY with the values written to their buffers. The MOD, IDLY, CHnDLYm, DACINTx, and POyDLY will take effect according to the LDMOD. After 1 is written to the LDOK field, the values in the buffers of above registers are not effective and the buffers cannot be written until the values in buffers are loaded into their internal registers. LDOK can be written only when PDBEN is set or it can be written at the same time with PDBEN being written to 1. It is automatically cleared when the values in buffers are loaded into the internal registers or the PDBEN is cleared. Writing 0 to it has no effect.

## 34.3.2 Modulus register (PDBx\_MOD)

Address: 4003\_6000h base + 4h offset = 4003\_6004h

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	0																1															
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	

### 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.
15–0 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.

### 34.3.3 Counter register (PDBx\_CNT)

Address: 4003\_6000h base + 8h offset = 4003\_6008h

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	0																0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### 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.
15–0 CNT	PDB Counter  Contains the current value of the counter.

### 34.3.4 Interrupt Delay register (PDBx\_IDLY)

Address: 4003\_6000h base + Ch offset = 4003\_600Ch

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	0																0															
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.
15–0 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

Table continues on the next page...

**PDBx\_IDLY field descriptions (continued)**

Field	Description
	equal to the IDLY. Reading this field returns the value of internal register that is effective for the current cycle of the PDB.

**34.3.5 Channel n Control register 1 (PDBx\_CHnC1)**

Each PDB channel has one control register, CHnC1. The bits in this register control the functionality of each PDB channel operation.

Address: 4003\_6000h base + 10h offset + (40d × i), where i=0d to 1d

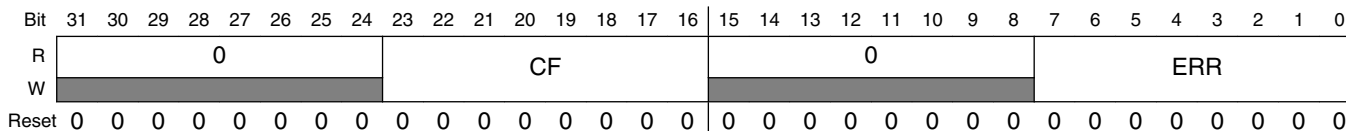
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								BB								TOS				EN											
W	0								0								0				0											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**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.
7–0 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.

### 34.3.6 Channel n Status register (PDBx\_CHnS)

Address: 4003\_6000h base + 14h offset + (40d × i), where i=0d to 1d

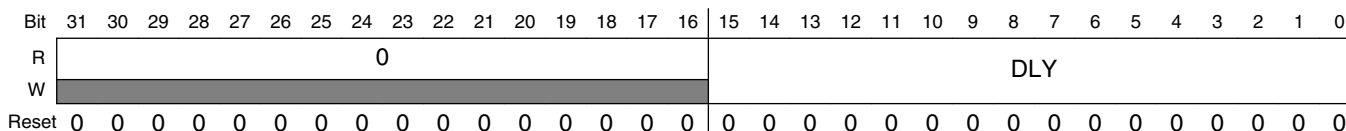


#### PDBx\_CHnS field descriptions

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] bit is set when the PDB counter matches the CHnDLYm. Write 0 to clear these bits.
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–0 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.

### 34.3.7 Channel n Delay 0 register (PDBx\_CHnDLY0)

Address: 4003\_6000h base + 18h offset + (40d × i), where i=0d to 1d



#### 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.
15–0 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.



### 34.3.8 Channel n Delay 1 register (PDBx\_CHnDLY1)

Address: 4003\_6000h base + 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	0																0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

#### PDBx\_CHnDLY1 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–0 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.

### 34.3.9 DAC Interval Trigger n Control register (PDBx\_DACINTCn)

Address: 4003\_6000h base + 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	0																
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	0															0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

#### PDBx\_DACINTCn 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  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.

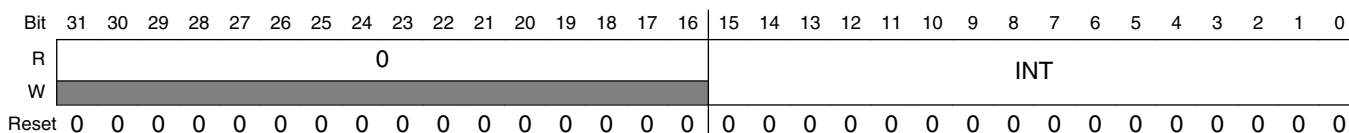
Table continues on the next page...

**PDBx\_DACINTCn field descriptions (continued)**

Field	Description
0 TOE	DAC Interval Trigger Enable  Enables the DAC interval trigger.  0 DAC interval trigger disabled. 1 DAC interval trigger enabled.

**34.3.10 DAC Interval n register (PDBx\_DACINTn)**

Address: 4003\_6000h base + 154h offset + (8d × i), where i=0d to 0d

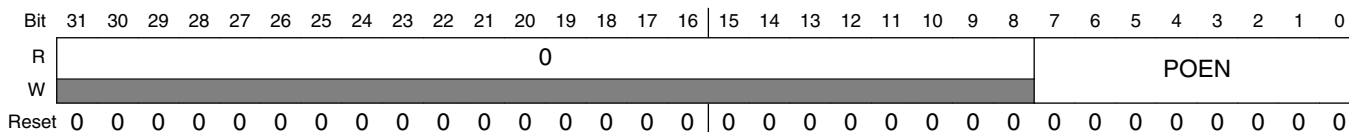


**PDBx\_DACINTn field descriptions**

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–0 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.

**34.3.11 Pulse-Out n Enable register (PDBx\_POEN)**

Address: 4003\_6000h base + 190h offset = 4003\_6190h



**PDBx\_POEN field descriptions**

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–0 POEN	PDB Pulse-Out Enable  Enables the pulse output. Only lower Y bits are implemented in this MCU.

Table continues on the next page...

**PDBx\_POEN field descriptions (continued)**

Field	Description
0	PDB Pulse-Out disabled
1	PDB Pulse-Out enabled

**34.3.12 Pulse-Out n Delay register (PDBx\_POnDLY)**

Address: 4003\_6000h base + 194h 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	DLY1																DLY2															
W	DLY1																DLY2															
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	

**PDBx\_POnDLY 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.
15–0 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.

**34.4 Functional description****34.4.1 PDB pre-trigger and trigger outputs**

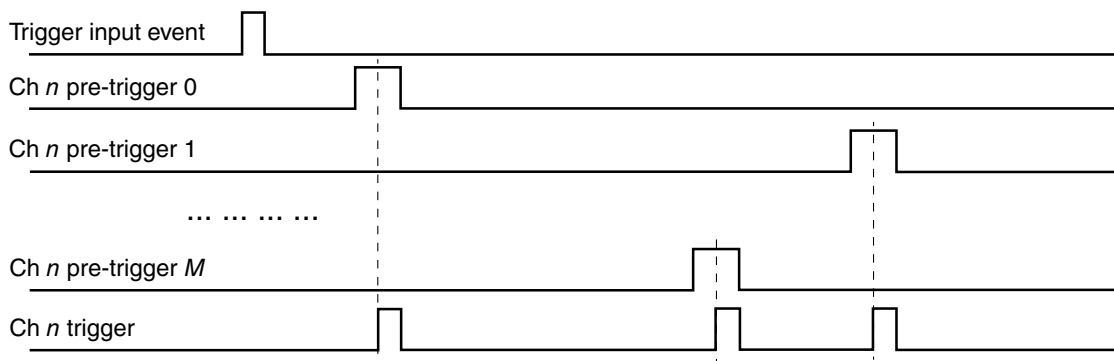
The PDB contains a counter whose output is compared to several different digital values. If the PDB is enabled, then a trigger input event will reset the counter and make it start to count. A trigger input event is defined as a rising edge being detected on a selected trigger input source, or if a software 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:

## Functional description

- Trigger input event to pre-trigger  $m = (\text{prescaler} \times \text{multiplication factor} \times \text{delay } m) + 2$  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  $\text{CHnDLYm}$  registers, and the pre-triggers can be enabled or disabled in  $\text{CHnC1[EN}[m]]$ .



**Figure 34-50. Pre-trigger and trigger outputs**

The delay in  $\text{CHnDLYm}$  register can be optionally bypassed, if  $\text{CHnC1[TOS}[m]]$  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  $\text{CHnC1[BB}[m]]$ , 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  $ADC_nSC1[COCO]$ ; the  $ADC_nSC1[COCO]$  should be cleared after the conversion result is read, so that the next rising edge of  $ADC_nSC1[COCO]$  can be generated to clear the lock later. The lock becomes inactive when:

- the rising edge of corresponding  $ADC_nSC1[COCO]$  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  $CH_nS[ERR[m]]$  (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 set in  $IDLY$  register, 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.

### 34.4.2 PDB trigger input source selection

The PDB has up to 15 trigger input sources, namely Trigger-In 14. They are connected to on-chip or off-chip event sources. The PDB can be triggered by software through  $SC[SWTRIG]$ .  $SC[TRIGSEL]$  selects the active trigger input source or software trigger.

For the trigger input sources implemented in this MCU, see chip configuration information.

### 34.4.3 Pulse-Out's

PDB can generate pulse outputs of configurable width. When PDB counter reaches the value set in  $POyDLY[DLY1]$ , the Pulse-Out goes high; when the counter reaches  $POyDLY[DLY2]$ , it goes low.  $POyDLY[DLY2]$  can be set either greater or less than  $POyDLY[DLY1]$ .

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.

### 34.4.4 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 as shown in the table below.

**Table 34-52. Circumstances of update to the delay registers**

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 the MOD register 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 the MOD register 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.

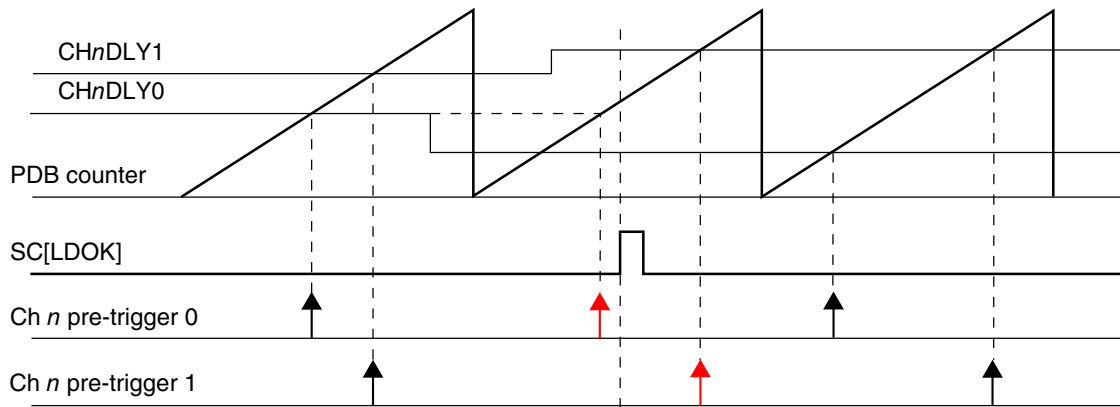


Figure 34-51. Registers update with SC[LDMOD] = 00

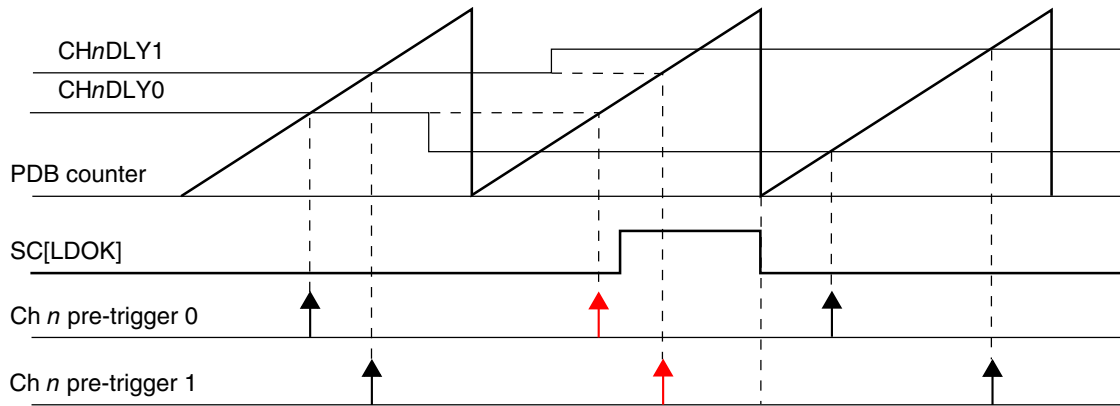


Figure 34-52. Registers update with SC[LDMOD] = x1

### 34.4.5 Interrupts

PDB can generate two interrupts: PDB interrupt and PDB sequence error interrupt. The following table summarizes the interrupts.

Table 34-53. PDB interrupt summary

Interrupt	Flags	Enable bit
PDB Interrupt	SC[PDBIF]	SC[PDBIE] = 1 and SC[DMAEN] = 0
PDB Sequence Error Interrupt	CHnS[ERRm]	SC[PDBEIE] = 1

### 34.4.6 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.

## 34.5 Application information

### 34.5.1 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 35

## FlexTimer Module (FTM)

### 35.1 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.

#### 35.1.1 FlexTimer philosophy

The FlexTimer is built upon a simple timer, the HCS08 Timer PWM Module – TPM, used for many years on Freescale's 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.

### 35.1.2 Features

The FTM features include:

- FTM source clock is selectable
  - 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
- 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

### 35.1.3 Modes of operation

When the MCU is in an active BDM mode, the FTM temporarily suspends all counting until the MCU 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 MCU from Wait mode, the power can then be saved by disabling FTM functions before entering Wait mode.

### 35.1.4 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.

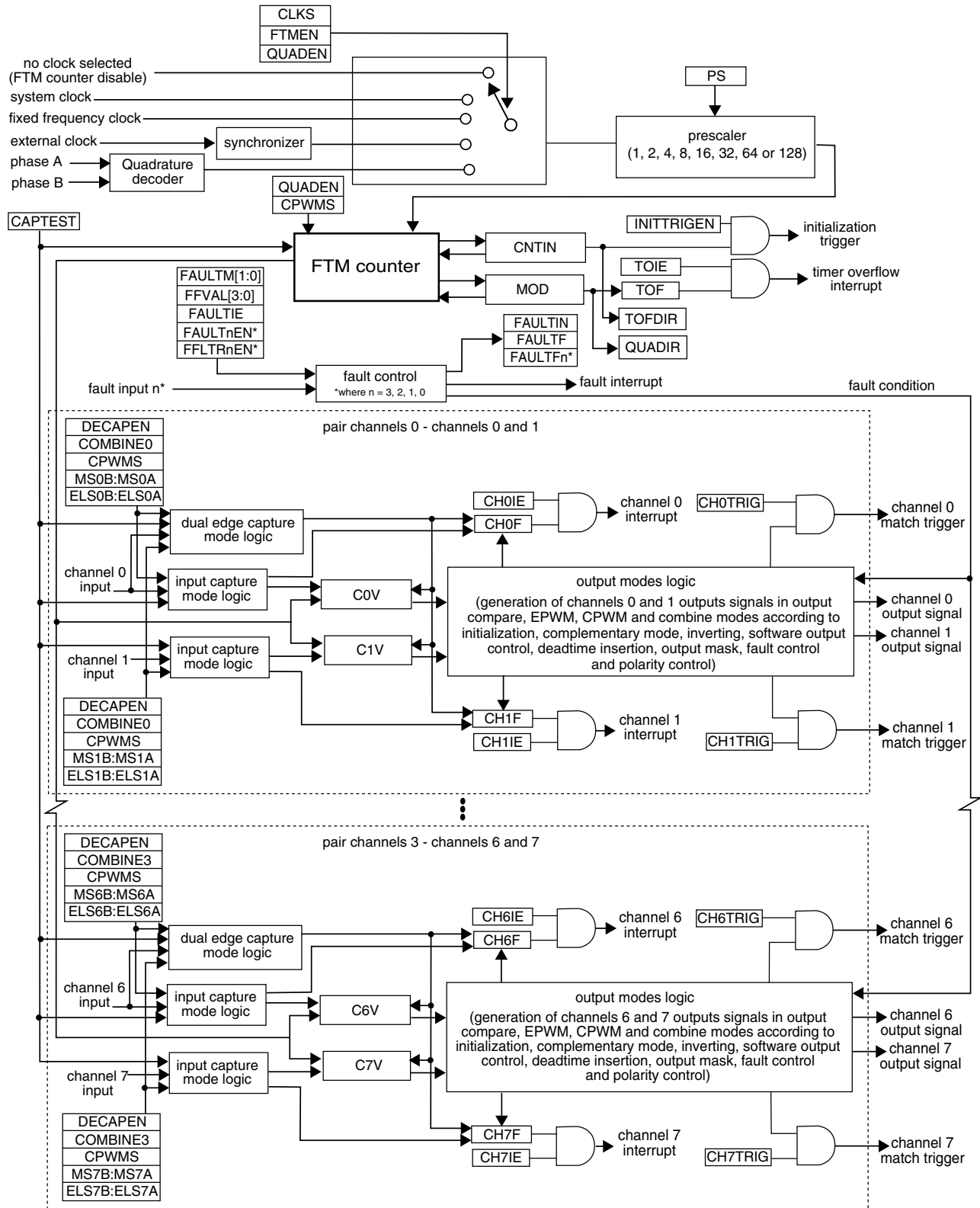


Figure 35-1. FTM block diagram

## 35.2 FTM signal descriptions

Table 35-1 shows the user-accessible signals for the FTM.

**Table 35-1. 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 <a href="#">Quadrature Decoder mode</a> .
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 <a href="#">Quadrature Decoder mode</a> .

## 35.3 Memory map and register definition

### 35.3.1 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.

**35.3.2 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
4003_8000	Status And Control (FTM0_SC)	32	R/W	0000_0000h	<a href="#">35.3.3/724</a>
4003_8004	Counter (FTM0_CNT)	32	R/W	0000_0000h	<a href="#">35.3.4/725</a>
4003_8008	Modulo (FTM0_MOD)	32	R/W	0000_0000h	<a href="#">35.3.5/726</a>
4003_800C	Channel (n) Status And Control (FTM0_C0SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_8010	Channel (n) Value (FTM0_C0V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_8014	Channel (n) Status And Control (FTM0_C1SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_8018	Channel (n) Value (FTM0_C1V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_801C	Channel (n) Status And Control (FTM0_C2SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_8020	Channel (n) Value (FTM0_C2V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_8024	Channel (n) Status And Control (FTM0_C3SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_8028	Channel (n) Value (FTM0_C3V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_802C	Channel (n) Status And Control (FTM0_C4SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_8030	Channel (n) Value (FTM0_C4V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_8034	Channel (n) Status And Control (FTM0_C5SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_8038	Channel (n) Value (FTM0_C5V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_803C	Channel (n) Status And Control (FTM0_C6SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_8040	Channel (n) Value (FTM0_C6V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_8044	Channel (n) Status And Control (FTM0_C7SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_8048	Channel (n) Value (FTM0_C7V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_804C	Counter Initial Value (FTM0_CNTIN)	32	R/W	0000_0000h	<a href="#">35.3.8/730</a>
4003_8050	Capture And Compare Status (FTM0_STATUS)	32	R/W	0000_0000h	<a href="#">35.3.9/731</a>
4003_8054	Features Mode Selection (FTM0_MODE)	32	R/W	0000_0004h	<a href="#">35.3.10/733</a>
4003_8058	Synchronization (FTM0_SYNC)	32	R/W	0000_0000h	<a href="#">35.3.11/735</a>
4003_805C	Initial State For Channels Output (FTM0_OUTINIT)	32	R/W	0000_0000h	<a href="#">35.3.12/737</a>
4003_8060	Output Mask (FTM0_OUTMASK)	32	R/W	0000_0000h	<a href="#">35.3.13/738</a>

Table continues on the next page...

## FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_8064	Function For Linked Channels (FTM0_COMBINE)	32	R/W	0000_0000h	<a href="#">35.3.14/740</a>
4003_8068	Deadtime Insertion Control (FTM0_DEADTIME)	32	R/W	0000_0000h	<a href="#">35.3.15/745</a>
4003_806C	FTM External Trigger (FTM0_EXTTRIG)	32	R/W	0000_0000h	<a href="#">35.3.16/746</a>
4003_8070	Channels Polarity (FTM0_POL)	32	R/W	0000_0000h	<a href="#">35.3.17/748</a>
4003_8074	Fault Mode Status (FTM0_FMS)	32	R/W	0000_0000h	<a href="#">35.3.18/750</a>
4003_8078	Input Capture Filter Control (FTM0_FILTER)	32	R/W	0000_0000h	<a href="#">35.3.19/752</a>
4003_807C	Fault Control (FTM0_FLTCTRL)	32	R/W	0000_0000h	<a href="#">35.3.20/753</a>
4003_8080	Quadrature Decoder Control And Status (FTM0_QDCTRL)	32	R/W	0000_0000h	<a href="#">35.3.21/756</a>
4003_8084	Configuration (FTM0_CONF)	32	R/W	0000_0000h	<a href="#">35.3.22/758</a>
4003_8088	FTM Fault Input Polarity (FTM0_FLTPOL)	32	R/W	0000_0000h	<a href="#">35.3.23/759</a>
4003_808C	Synchronization Configuration (FTM0_SYNCONF)	32	R/W	0000_0000h	<a href="#">35.3.24/760</a>
4003_8090	FTM Inverting Control (FTM0_INVCTRL)	32	R/W	0000_0000h	<a href="#">35.3.25/762</a>
4003_8094	FTM Software Output Control (FTM0_SWOCTRL)	32	R/W	0000_0000h	<a href="#">35.3.26/763</a>
4003_8098	FTM PWM Load (FTM0_PWMLOAD)	32	R/W	0000_0000h	<a href="#">35.3.27/766</a>
4003_9000	Status And Control (FTM1_SC)	32	R/W	0000_0000h	<a href="#">35.3.3/724</a>
4003_9004	Counter (FTM1_CNT)	32	R/W	0000_0000h	<a href="#">35.3.4/725</a>
4003_9008	Modulo (FTM1_MOD)	32	R/W	0000_0000h	<a href="#">35.3.5/726</a>
4003_900C	Channel (n) Status And Control (FTM1_C0SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_9010	Channel (n) Value (FTM1_C0V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_9014	Channel (n) Status And Control (FTM1_C1SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_9018	Channel (n) Value (FTM1_C1V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_901C	Channel (n) Status And Control (FTM1_C2SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_9020	Channel (n) Value (FTM1_C2V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_9024	Channel (n) Status And Control (FTM1_C3SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_9028	Channel (n) Value (FTM1_C3V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_902C	Channel (n) Status And Control (FTM1_C4SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_9030	Channel (n) Value (FTM1_C4V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_9034	Channel (n) Status And Control (FTM1_C5SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>

Table continues on the next page...



## FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_9038	Channel (n) Value (FTM1_C5V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_903C	Channel (n) Status And Control (FTM1_C6SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_9040	Channel (n) Value (FTM1_C6V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_9044	Channel (n) Status And Control (FTM1_C7SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_9048	Channel (n) Value (FTM1_C7V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_904C	Counter Initial Value (FTM1_CNTIN)	32	R/W	0000_0000h	<a href="#">35.3.8/730</a>
4003_9050	Capture And Compare Status (FTM1_STATUS)	32	R/W	0000_0000h	<a href="#">35.3.9/731</a>
4003_9054	Features Mode Selection (FTM1_MODE)	32	R/W	0000_0004h	<a href="#">35.3.10/733</a>
4003_9058	Synchronization (FTM1_SYNC)	32	R/W	0000_0000h	<a href="#">35.3.11/735</a>
4003_905C	Initial State For Channels Output (FTM1_OUTINIT)	32	R/W	0000_0000h	<a href="#">35.3.12/737</a>
4003_9060	Output Mask (FTM1_OUTMASK)	32	R/W	0000_0000h	<a href="#">35.3.13/738</a>
4003_9064	Function For Linked Channels (FTM1_COMBINE)	32	R/W	0000_0000h	<a href="#">35.3.14/740</a>
4003_9068	Deadtime Insertion Control (FTM1_DEADTIME)	32	R/W	0000_0000h	<a href="#">35.3.15/745</a>
4003_906C	FTM External Trigger (FTM1_EXTTRIG)	32	R/W	0000_0000h	<a href="#">35.3.16/746</a>
4003_9070	Channels Polarity (FTM1_POL)	32	R/W	0000_0000h	<a href="#">35.3.17/748</a>
4003_9074	Fault Mode Status (FTM1_FMS)	32	R/W	0000_0000h	<a href="#">35.3.18/750</a>
4003_9078	Input Capture Filter Control (FTM1_FILTER)	32	R/W	0000_0000h	<a href="#">35.3.19/752</a>
4003_907C	Fault Control (FTM1_FLTCTRL)	32	R/W	0000_0000h	<a href="#">35.3.20/753</a>
4003_9080	Quadrature Decoder Control And Status (FTM1_QDCTRL)	32	R/W	0000_0000h	<a href="#">35.3.21/756</a>
4003_9084	Configuration (FTM1_CONF)	32	R/W	0000_0000h	<a href="#">35.3.22/758</a>
4003_9088	FTM Fault Input Polarity (FTM1_FLTPOL)	32	R/W	0000_0000h	<a href="#">35.3.23/759</a>
4003_908C	Synchronization Configuration (FTM1_SYNCONF)	32	R/W	0000_0000h	<a href="#">35.3.24/760</a>
4003_9090	FTM Inverting Control (FTM1_INVCTRL)	32	R/W	0000_0000h	<a href="#">35.3.25/762</a>
4003_9094	FTM Software Output Control (FTM1_SWOCTRL)	32	R/W	0000_0000h	<a href="#">35.3.26/763</a>
4003_9098	FTM PWM Load (FTM1_PWMLOAD)	32	R/W	0000_0000h	<a href="#">35.3.27/766</a>

## FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_A000	Status And Control (FTM2_SC)	32	R/W	0000_0000h	<a href="#">35.3.3/724</a>
4003_A004	Counter (FTM2_CNT)	32	R/W	0000_0000h	<a href="#">35.3.4/725</a>
4003_A008	Modulo (FTM2_MOD)	32	R/W	0000_0000h	<a href="#">35.3.5/726</a>
4003_A00C	Channel (n) Status And Control (FTM2_C0SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_A010	Channel (n) Value (FTM2_C0V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_A014	Channel (n) Status And Control (FTM2_C1SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_A018	Channel (n) Value (FTM2_C1V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_A01C	Channel (n) Status And Control (FTM2_C2SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_A020	Channel (n) Value (FTM2_C2V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_A024	Channel (n) Status And Control (FTM2_C3SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_A028	Channel (n) Value (FTM2_C3V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_A02C	Channel (n) Status And Control (FTM2_C4SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_A030	Channel (n) Value (FTM2_C4V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_A034	Channel (n) Status And Control (FTM2_C5SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_A038	Channel (n) Value (FTM2_C5V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_A03C	Channel (n) Status And Control (FTM2_C6SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_A040	Channel (n) Value (FTM2_C6V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_A044	Channel (n) Status And Control (FTM2_C7SC)	32	R/W	0000_0000h	<a href="#">35.3.6/727</a>
4003_A048	Channel (n) Value (FTM2_C7V)	32	R/W	0000_0000h	<a href="#">35.3.7/730</a>
4003_A04C	Counter Initial Value (FTM2_CNTIN)	32	R/W	0000_0000h	<a href="#">35.3.8/730</a>
4003_A050	Capture And Compare Status (FTM2_STATUS)	32	R/W	0000_0000h	<a href="#">35.3.9/731</a>
4003_A054	Features Mode Selection (FTM2_MODE)	32	R/W	0000_0004h	<a href="#">35.3.10/733</a>
4003_A058	Synchronization (FTM2_SYNC)	32	R/W	0000_0000h	<a href="#">35.3.11/735</a>
4003_A05C	Initial State For Channels Output (FTM2_OUTINIT)	32	R/W	0000_0000h	<a href="#">35.3.12/737</a>
4003_A060	Output Mask (FTM2_OUTMASK)	32	R/W	0000_0000h	<a href="#">35.3.13/738</a>
4003_A064	Function For Linked Channels (FTM2_COMBINE)	32	R/W	0000_0000h	<a href="#">35.3.14/740</a>
4003_A068	Deadtime Insertion Control (FTM2_DEADTIME)	32	R/W	0000_0000h	<a href="#">35.3.15/745</a>
4003_A06C	FTM External Trigger (FTM2_EXTTRIG)	32	R/W	0000_0000h	<a href="#">35.3.16/746</a>
4003_A070	Channels Polarity (FTM2_POL)	32	R/W	0000_0000h	<a href="#">35.3.17/748</a>
4003_A074	Fault Mode Status (FTM2_FMS)	32	R/W	0000_0000h	<a href="#">35.3.18/750</a>
4003_A078	Input Capture Filter Control (FTM2_FILTER)	32	R/W	0000_0000h	<a href="#">35.3.19/752</a>

Table continues on the next page...

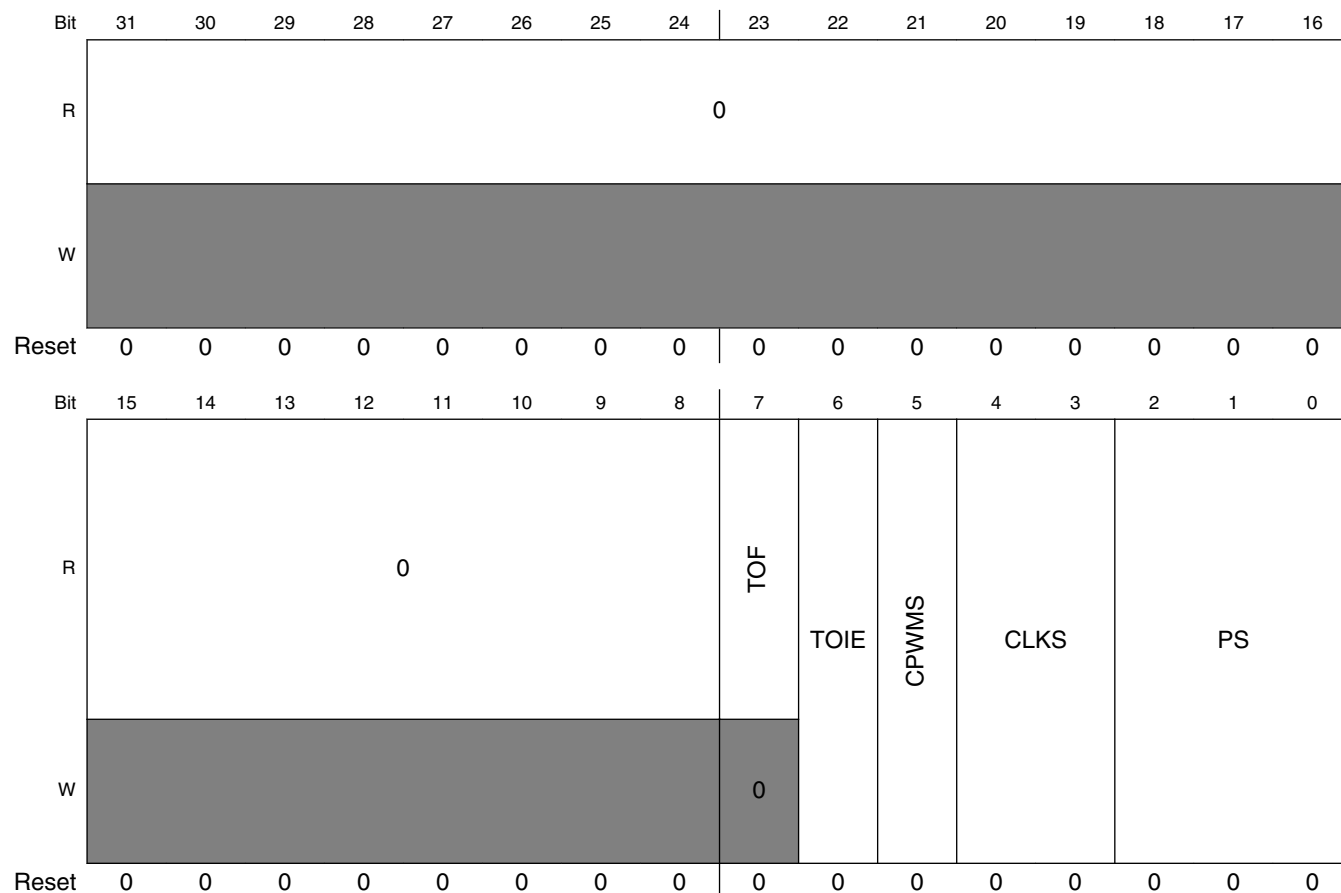
## FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_A07C	Fault Control (FTM2_FLTCTRL)	32	R/W	0000_0000h	<a href="#">35.3.20/753</a>
4003_A080	Quadrature Decoder Control And Status (FTM2_QDCTRL)	32	R/W	0000_0000h	<a href="#">35.3.21/756</a>
4003_A084	Configuration (FTM2_CONF)	32	R/W	0000_0000h	<a href="#">35.3.22/758</a>
4003_A088	FTM Fault Input Polarity (FTM2_FLTPOL)	32	R/W	0000_0000h	<a href="#">35.3.23/759</a>
4003_A08C	Synchronization Configuration (FTM2_SYNCONF)	32	R/W	0000_0000h	<a href="#">35.3.24/760</a>
4003_A090	FTM Inverting Control (FTM2_INVCTRL)	32	R/W	0000_0000h	<a href="#">35.3.25/762</a>
4003_A094	FTM Software Output Control (FTM2_SWOCTRL)	32	R/W	0000_0000h	<a href="#">35.3.26/763</a>
4003_A098	FTM PWM Load (FTM2_PWMLOAD)	32	R/W	0000_0000h	<a href="#">35.3.27/766</a>

### 35.3.3 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	<p>Timer Overflow Flag</p> <p>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.</p> <p>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.</p> <p>0 FTM counter has not overflowed. 1 FTM counter has overflowed.</p>

Table continues on the next page...

## FTMx\_SC field descriptions (continued)

Field	Description
6 TOIE	<p>Timer Overflow Interrupt Enable</p> <p>Enables FTM overflow interrupts.</p> <p>0 Disable TOF interrupts. Use software polling. 1 Enable TOF interrupts. An interrupt is generated when TOF equals one.</p>
5 CPWMS	<p>Center-Aligned PWM Select</p> <p>Selects CPWM mode. This mode configures the FTM to operate in Up-Down Counting mode.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 FTM counter operates in Up Counting mode. 1 FTM counter operates in Up-Down Counting mode.</p>
4–3 CLKS	<p>Clock Source Selection</p> <p>Selects one of the three FTM counter clock sources.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>00 No clock selected. This in effect disables the FTM counter. 01 System clock 10 Fixed frequency clock 11 External clock</p>
2–0 PS	<p>Prescale Factor Selection</p> <p>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.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>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</p>

### 35.3.4 Counter (FTMx\_CNT)

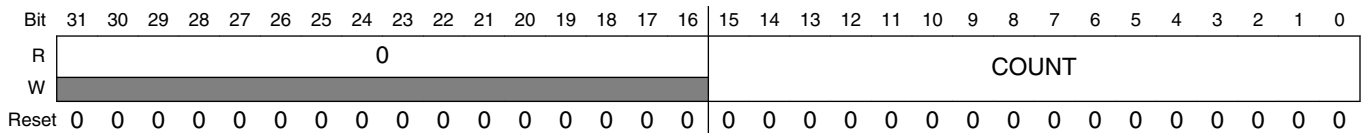
The CNT register contains the FTM counter value.

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.

## Memory map and register definition

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.
15–0 COUNT	Counter Value

## 35.3.5 Modulo (FTMx\_MOD)

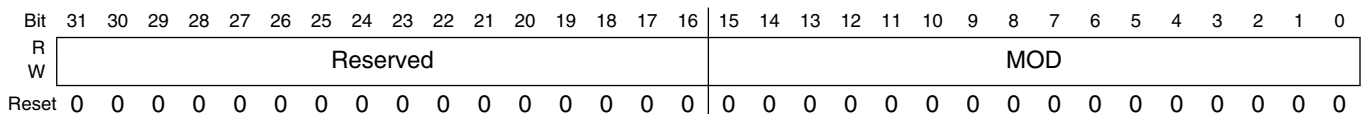
The Modulo register contains the modulo value for the FTM counter. After the FTM counter reaches the modulo value, the overflow flag (TOF) becomes set at the next clock, and the next value of FTM counter depends on the selected counting method; see [Counter](#).

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.
15–0 MOD	Modulo Value

### 35.3.6 Channel (n) Status And Control (FTMx\_CnSC)

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

**Table 35-67. 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	Output Compare	01	Toggle Output on match		
						10	Clear Output on match		
						11	Set Output on match		
			1X	Edge-Aligned PWM	10	High-true pulses (clear Output on match)			
					X1	Low-true pulses (set Output on match)			
			1	XX	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	XX	10	Combine PWM	High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
									X1

Table continues on the next page...

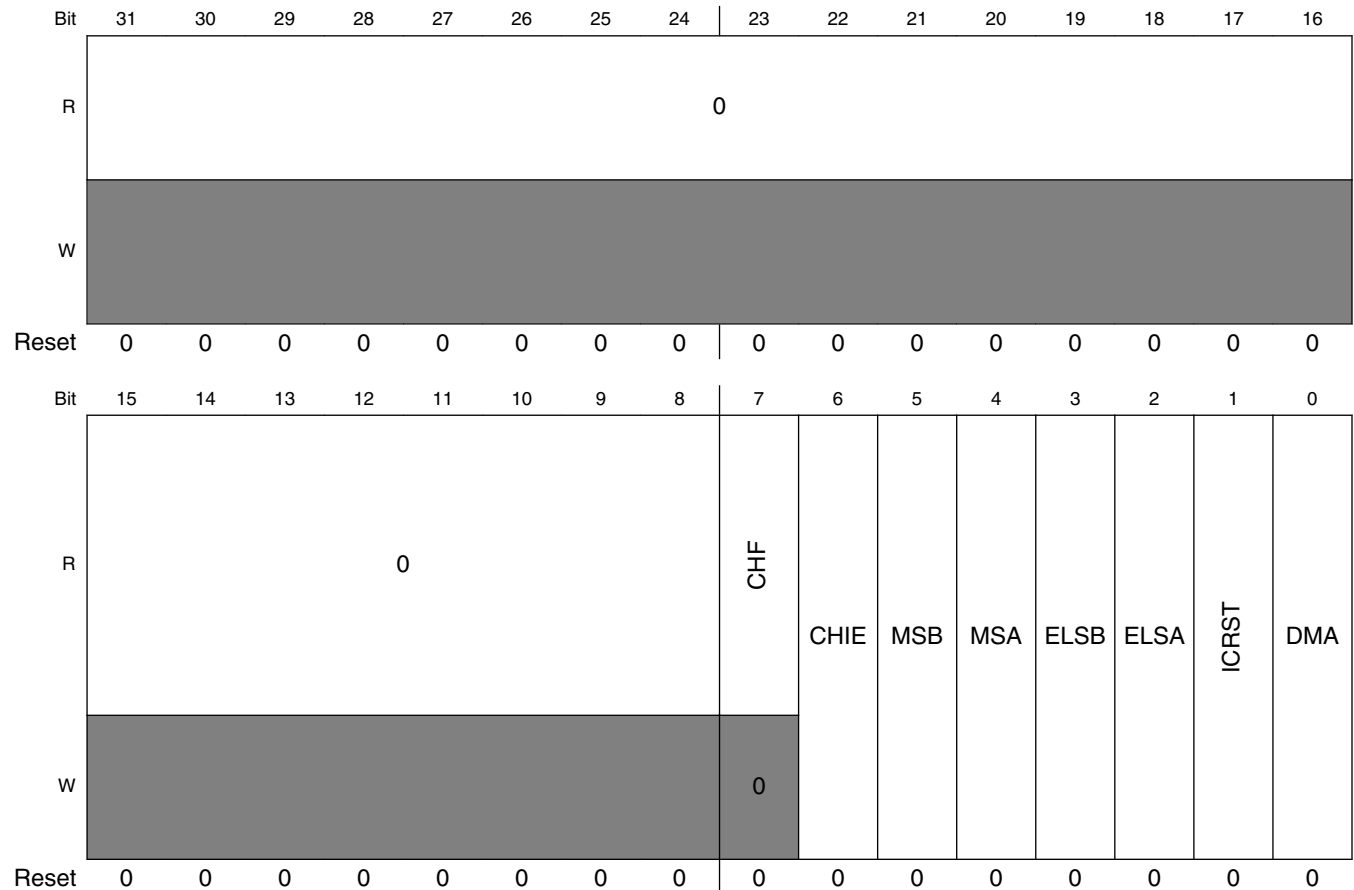
**Table 35-67. Mode, edge, and level selection (continued)**

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
1	0	0	X0	See the following table (Table 35-8).	Dual Edge Capture	One-Shot Capture mode
			X1			Continuous Capture mode

**Table 35-68. 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





## FTMx\_CnSC 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 <a href="#">Table 35-7</a> .  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 <a href="#">Table 35-7</a> .  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 <a href="#">Table 35-7</a> .  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 <a href="#">Table 35-7</a> .  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.

### 35.3.7 Channel (n) Value (FTMx\_CnV)

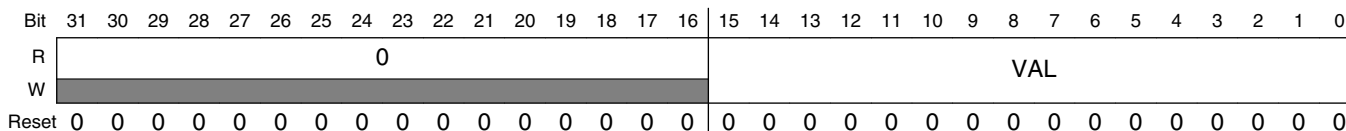
These registers contain the captured FTM counter value for the input modes or the match value for the output modes.

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 FTMMEN = 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



#### 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.
15–0 VAL	Channel Value  Captured FTM counter value of the input modes or the match value for the output modes

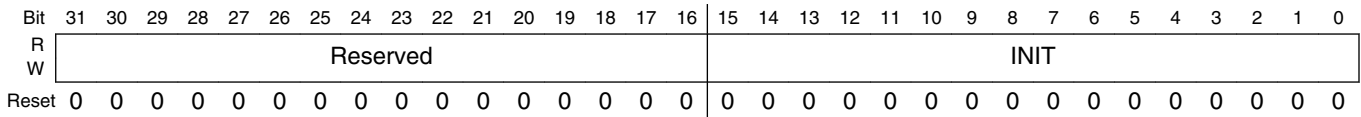
### 35.3.8 Counter Initial Value (FTMx\_CNTIN)

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



**FTMx\_CNTIN field descriptions**

Field	Description
31–16 Reserved	This field is reserved.
15–0 INIT	Initial Value Of The FTM Counter

**35.3.9 Capture And Compare Status (FTMx\_STATUS)**

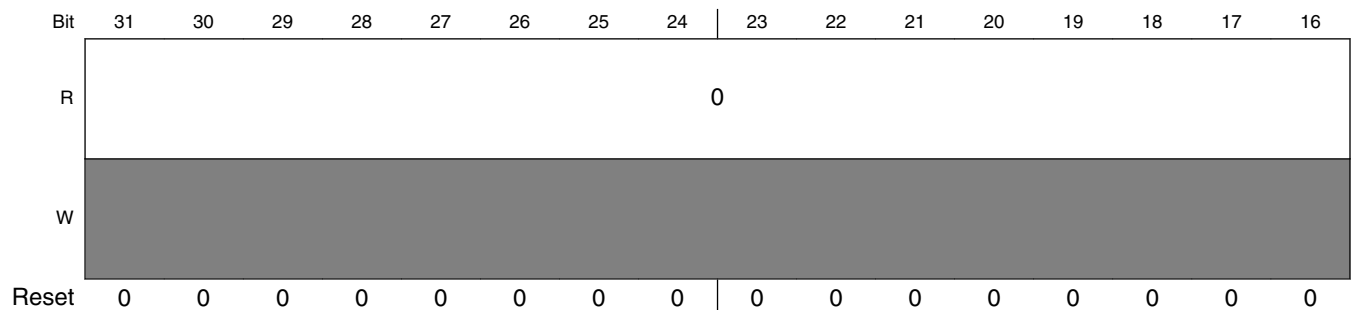
The STATUS register contains a copy of the status flag CHnF bit in CnSC for each FTM channel for software convenience.

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



## Memory map and register definition

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.

**35.3.10 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 <a href="#">PWM synchronization</a> . 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.

### 35.3.11 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

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

## Memory map and register definition

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SWSYNC	TRIG2	TRIG1	TRIG0	SYNCHOM	REINIT	CNTMAX	CNTMIN
W	0								0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### 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 ( <a href="#">FTM counter synchronization</a> )  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	<p>Maximum Loading Point Enable</p> <p>Selects the maximum loading point to PWM synchronization. See <a href="#">Boundary cycle and loading points</a>. If CNTMAX is 1, the selected loading point is when the FTM counter reaches its maximum value (MOD register).</p> <p>0 The maximum loading point is disabled. 1 The maximum loading point is enabled.</p>
0 CNTMIN	<p>Minimum Loading Point Enable</p> <p>Selects the minimum loading point to PWM synchronization. See <a href="#">Boundary cycle and loading points</a>. If CNTMIN is one, the selected loading point is when the FTM counter reaches its minimum value (CNTIN register).</p> <p>0 The minimum loading point is disabled. 1 The minimum loading point is enabled.</p>

## 35.3.12 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	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
7 CH7OI	<p>Channel 7 Output Initialization Value</p> <p>Selects the value that is forced into the channel output when the initialization occurs.</p> <p>0 The initialization value is 0. 1 The initialization value is 1.</p>
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.

**35.3.13 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

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								CH7OM	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM	CH0OM
W									CH7OM	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM	CH0OM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**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.

**35.3.14 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	DECAPO	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:ELSnA bits in Dual Edge Capture mode according to <a href="#">Table 35-7</a> . 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	<p>Combine Channels For <math>n = 6</math></p> <p>Enables the combine feature for channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when <math>MODE[WPDIS] = 1</math>.</p> <p>0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.</p>
23 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
22 FAULTEN2	<p>Fault Control Enable For <math>n = 4</math></p> <p>Enables the fault control in channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when <math>MODE[WPDIS] = 1</math>.</p> <p>0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.</p>
21 SYNCEN2	<p>Synchronization Enable For <math>n = 4</math></p> <p>Enables PWM synchronization of registers <math>C(n)V</math> and <math>C(n+1)V</math>.</p> <p>0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.</p>
20 DTEN2	<p>Deadtime Enable For <math>n = 4</math></p> <p>Enables the deadtime insertion in the channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when <math>MODE[WPDIS] = 1</math>.</p> <p>0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.</p>
19 DECAP2	<p>Dual Edge Capture Mode Captures For <math>n = 4</math></p> <p>Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.</p> <p>This field applies only when <math>DECAPEN = 1</math>.</p> <p>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.</p> <p>0 The dual edge captures are inactive. 1 The dual edge captures are active.</p>
18 DECAPEN2	<p>Dual Edge Capture Mode Enable For <math>n = 4</math></p> <p>Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of <math>MSnA</math>, <math>ELSnB:ELSnA</math> and <math>ELSn+1B:ELSn+1A</math> bits in Dual Edge Capture mode according to <a href="#">Table 35-7</a>.</p> <p>This field is write protected. It can be written only when <math>MODE[WPDIS] = 1</math>.</p> <p>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.</p>
17 COMP2	<p>Complement Of Channel (n) For <math>n = 4</math></p>

Table continues on the next page...

## FTMx\_COMBINE field descriptions (continued)

Field	Description
	<p>Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>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.</p>
16 COMBINE2	<p>Combine Channels For n = 4</p> <p>Enables the combine feature for channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.</p>
15 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
14 FAULTEN1	<p>Fault Control Enable For n = 2</p> <p>Enables the fault control in channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.</p>
13 SYNCEN1	<p>Synchronization Enable For n = 2</p> <p>Enables PWM synchronization of registers C(n)V and C(n+1)V.</p> <p>0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.</p>
12 DTEN1	<p>Deadtime Enable For n = 2</p> <p>Enables the deadtime insertion in the channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.</p>
11 DECAP1	<p>Dual Edge Capture Mode Captures For n = 2</p> <p>Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.</p> <p>This field applies only when DECAPEN = 1.</p> <p>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.</p> <p>0 The dual edge captures are inactive. 1 The dual edge captures are active.</p>
10 DECAPEN1	<p>Dual Edge Capture Mode Enable For n = 2</p> <p>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 <a href="#">Table 35-7</a>.</p>

Table continues on the next page...

**FTMx\_COMBINE field descriptions (continued)**

Field	Description
	<p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>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.</p>
9 COMP1	<p>Complement Of Channel (n) For n = 2</p> <p>Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>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.</p>
8 COMBINE1	<p>Combine Channels For n = 2</p> <p>Enables the combine feature for channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.</p>
7 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
6 FAULTEN0	<p>Fault Control Enable For n = 0</p> <p>Enables the fault control in channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.</p>
5 SYNCEN0	<p>Synchronization Enable For n = 0</p> <p>Enables PWM synchronization of registers C(n)V and C(n+1)V.</p> <p>0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.</p>
4 DTEN0	<p>Deadtime Enable For n = 0</p> <p>Enables the deadtime insertion in the channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.</p>
3 DECAPO	<p>Dual Edge Capture Mode Captures For n = 0</p> <p>Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits.</p> <p>This field applies only when DECAPEN = 1.</p> <p>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.</p> <p>0 The dual edge captures are inactive. 1 The dual edge captures are active.</p>

*Table continues on the next page...*



## FTMx\_COMBINE field descriptions (continued)

Field	Description
2 DECAPEN0	<p>Dual Edge Capture Mode Enable For n = 0</p> <p>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 <a href="#">Table 35-7</a>.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>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.</p>
1 COMP0	<p>Complement Of Channel (n) For n = 0</p> <p>Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>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.</p>
0 COMBINE0	<p>Combine Channels For n = 0</p> <p>Enables the combine feature for channels (n) and (n+1).</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.</p>

## 35.3.15 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	0																0		0													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## FTMx\_DEADTIME field descriptions

Field	Description
31–8 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
7–6 DTPS	<p>Deadtime Prescaler Value</p> <p>Selects the division factor of the system clock. This prescaled clock is used by the deadtime counter.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0x Divide the system clock by 1.</p>

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.
5-0 DTVVAL	<p>Deadtime Value</p> <p>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.</p> <p>Deadtime insert value = (DTPS × DTVVAL).</p> <p>DTVVAL selects the number of deadtime counts inserted as follows:</p> <p>When DTVVAL is 0, no counts are inserted.</p> <p>When DTVVAL is 1, 1 count is inserted.</p> <p>When DTVVAL is 2, 2 counts are inserted.</p> <p>This pattern continues up to a possible 63 counts.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p>

**35.3.16 FTM External Trigger (FTMx\_EXTTRIG)**

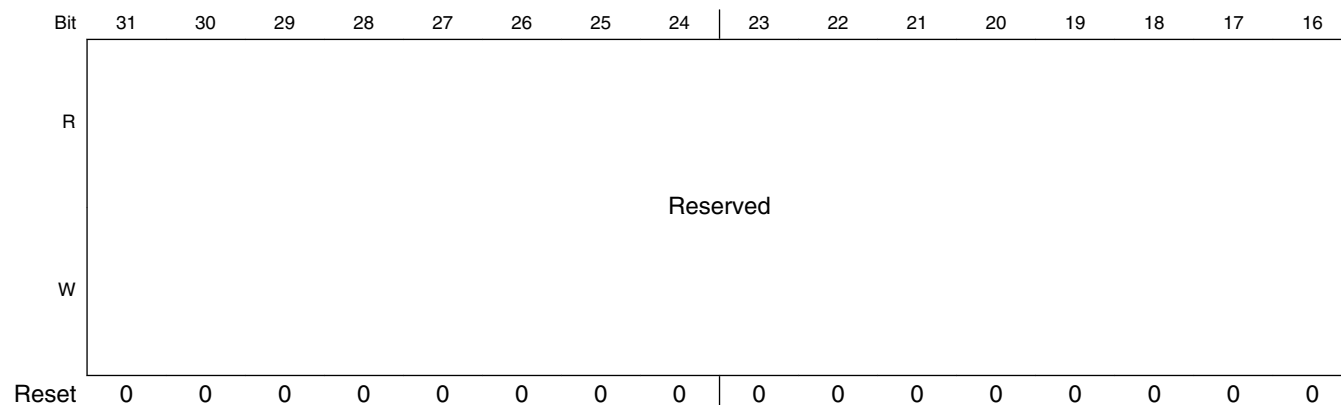
This register:

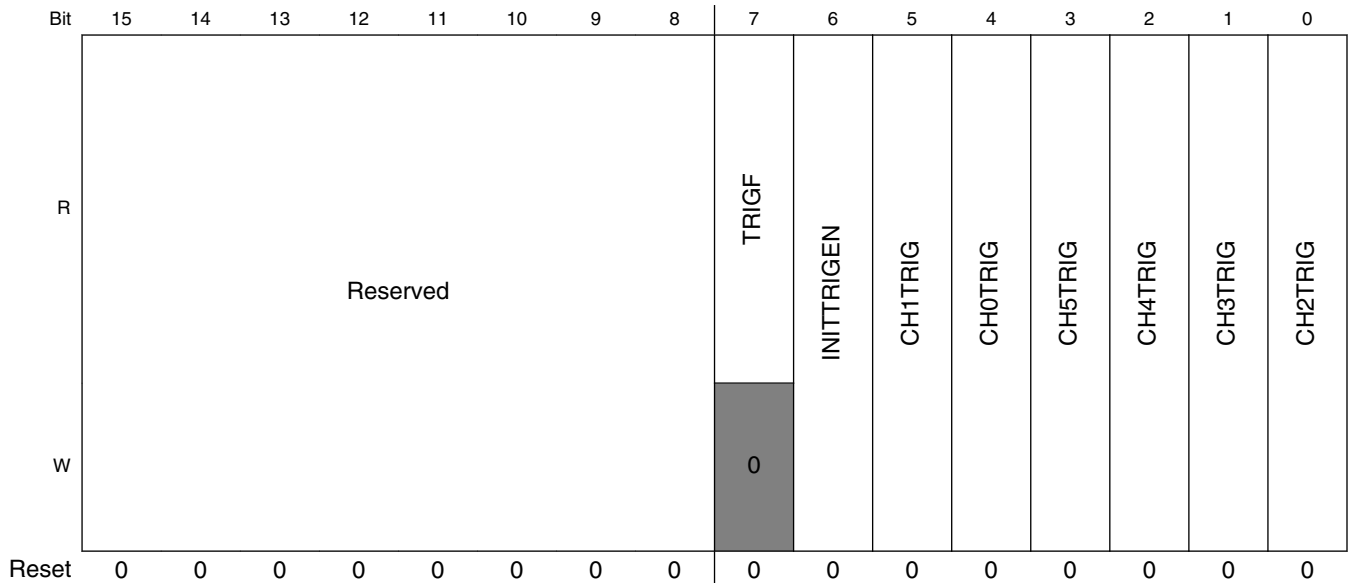
- Indicates when a channel trigger was generated
- Enables the generation of a trigger when the FTM counter is equal to its initial value
- Selects which channels are used in the generation of the channel triggers

Several channels can be selected to generate multiple triggers in one PWM period.

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	<p>Channel Trigger Flag</p> <p>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.</p> <p>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.</p> <p>0 No channel trigger was generated. 1 A channel trigger was generated.</p>
6 INITTRIGEN	<p>Initialization Trigger Enable</p> <p>Enables the generation of the trigger when the FTM counter is equal to the CNTIN register.</p> <p>0 The generation of initialization trigger is disabled. 1 The generation of initialization trigger is enabled.</p>
5 CH1TRIG	<p>Channel 1 Trigger Enable</p> <p>Enables the generation of the channel trigger when the FTM counter is equal to the CnV register.</p> <p>0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.</p>
4 CH0TRIG	<p>Channel 0 Trigger Enable</p> <p>Enables the generation of the channel trigger when the FTM counter is equal to the CnV register.</p> <p>0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.</p>
3 CH5TRIG	<p>Channel 5 Trigger Enable</p>

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.

**35.3.17 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	Reserved								POL7	POL6	POL5	POL4	POL3	POL2	POL1	POL0
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...*

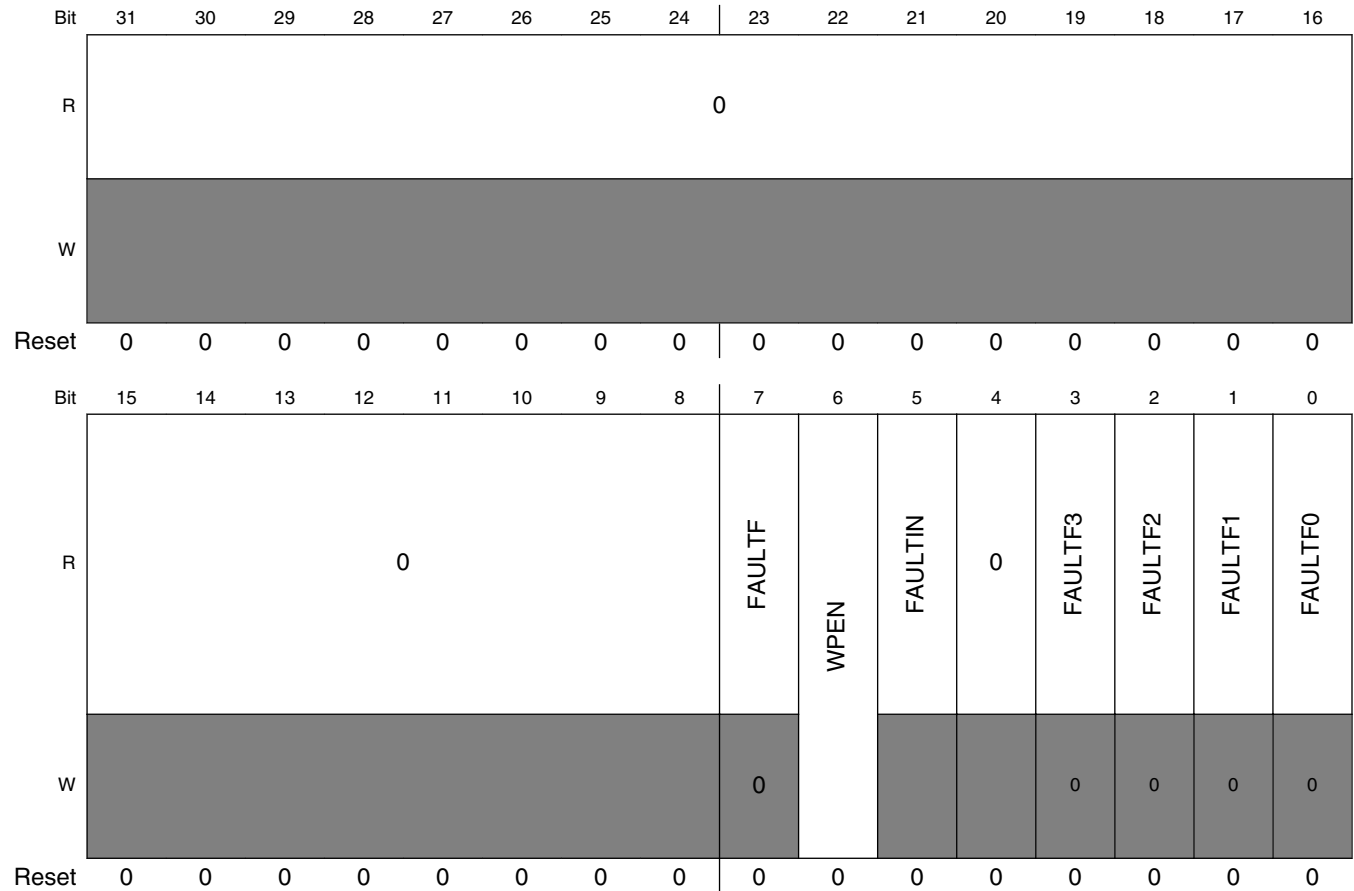
**FTMx\_POL field descriptions (continued)**

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.

**35.3.18 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



## 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	<p>Fault Detection Flag</p> <p>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.</p> <p>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.</p> <p>0 No fault condition was detected. 1 A fault condition was detected.</p>
6 WPEN	<p>Write Protection Enable</p> <p>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.</p> <p>0 Write protection is disabled. Write protected bits can be written. 1 Write protection is enabled. Write protected bits cannot be written.</p>
5 FAULTIN	<p>Fault Inputs</p> <p>Represents the logic OR of the enabled fault inputs after their filter (if their filter is enabled) when fault control is enabled.</p> <p>0 The logic OR of the enabled fault inputs is 0. 1 The logic OR of the enabled fault inputs is 1.</p>
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FAULTF3	<p>Fault Detection Flag 3</p> <p>Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input.</p> <p>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.</p> <p>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.</p> <p>0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.</p>
2 FAULTF2	<p>Fault Detection Flag 2</p> <p>Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input.</p> <p>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.</p>

*Table continues on the next page...*

**FTMx\_FMS field descriptions (continued)**

Field	Description
	<p>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.</p> <p>0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.</p>
1 FAULTF1	<p>Fault Detection Flag 1</p> <p>Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input.</p> <p>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.</p> <p>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.</p> <p>0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.</p>
0 FAULTF0	<p>Fault Detection Flag 0</p> <p>Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input.</p> <p>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.</p> <p>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.</p> <p>0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.</p>

**35.3.19 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	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	Reserved																CH3FVAL				CH2FVAL				CH1FVAL				CH0FVAL			
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\_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.
3–0 CH0FVAL	Channel 0 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.

**35.3.20 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	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				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. <b>NOTE:</b> 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...

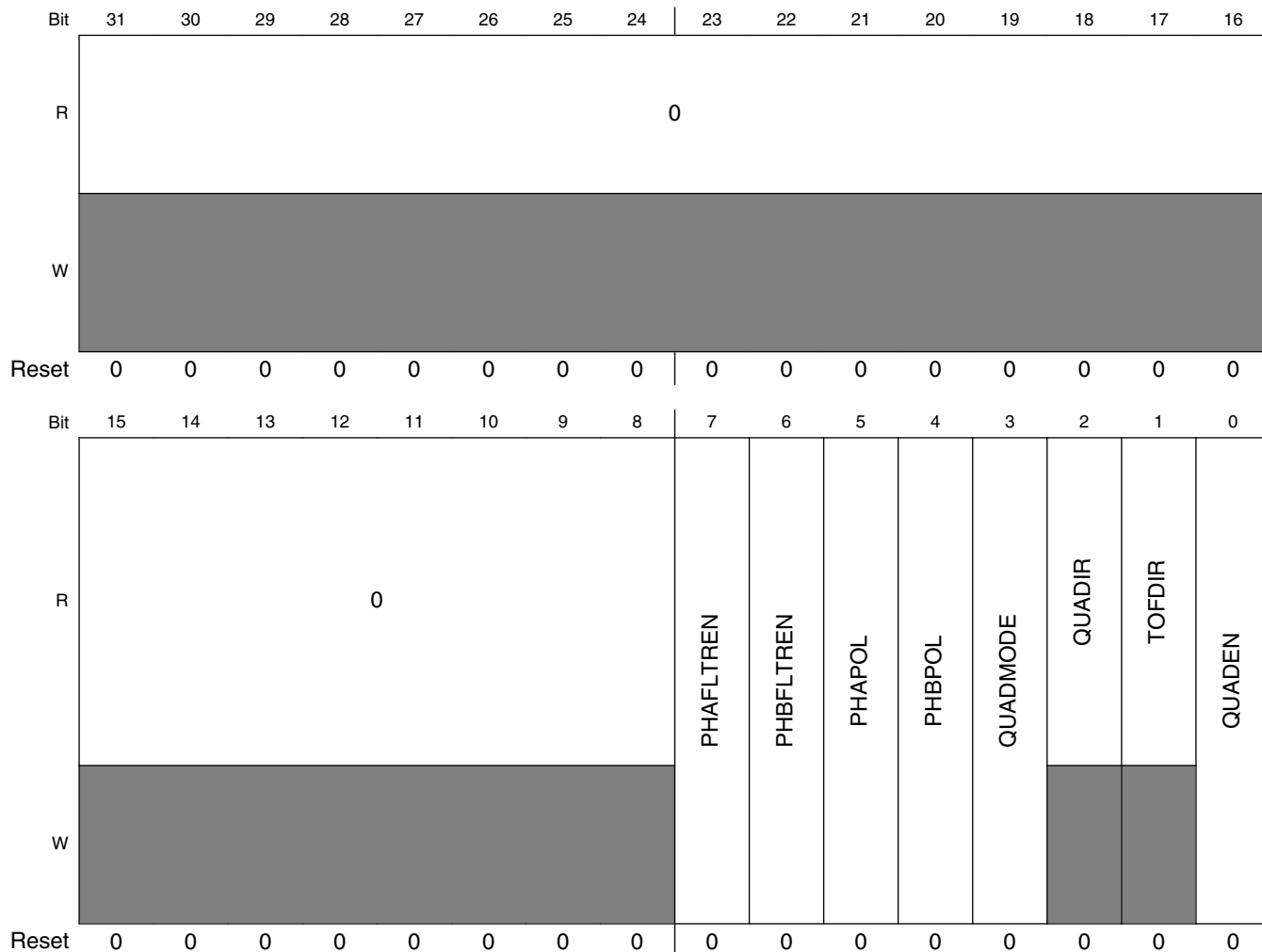
**FTMx\_FLTCTRL field descriptions (continued)**

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.

### 35.3.21 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...

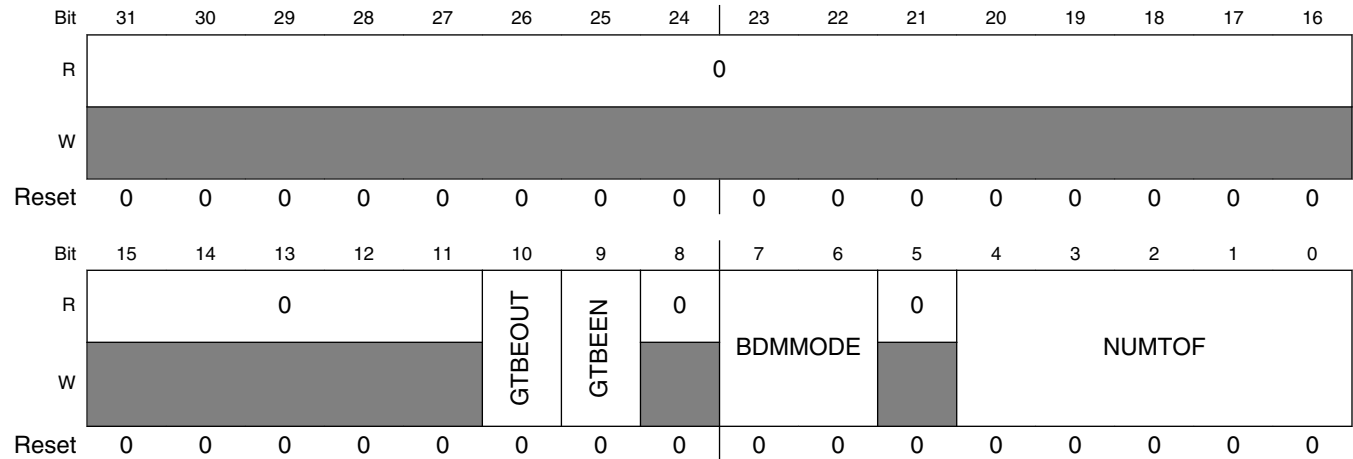
## FTMx\_QDCTRL field descriptions (continued)

Field	Description
6 PHBFLTREN	<p>Phase B Input Filter Enable</p> <p>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.</p> <p>0 Phase B input filter is disabled. 1 Phase B input filter is enabled.</p>
5 PHAPOL	<p>Phase A Input Polarity</p> <p>Selects the polarity for the quadrature decoder phase A input.</p> <p>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.</p>
4 PHBPOL	<p>Phase B Input Polarity</p> <p>Selects the polarity for the quadrature decoder phase B input.</p> <p>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.</p>
3 QUADMODE	<p>Quadrature Decoder Mode</p> <p>Selects the encoding mode used in the Quadrature Decoder mode.</p> <p>0 Phase A and phase B encoding mode. 1 Count and direction encoding mode.</p>
2 QUADIR	<p>FTM Counter Direction In Quadrature Decoder Mode</p> <p>Indicates the counting direction.</p> <p>0 Counting direction is decreasing (FTM counter decrement). 1 Counting direction is increasing (FTM counter increment).</p>
1 TOFDIR	<p>Timer Overflow Direction In Quadrature Decoder Mode</p> <p>Indicates if the TOF bit was set on the top or the bottom of counting.</p> <p>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).</p>
0 QUADEN	<p>Quadrature Decoder Mode Enable</p> <p>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 <a href="#">Table 35-7</a>.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 Quadrature Decoder mode is disabled. 1 Quadrature Decoder mode is enabled.</p>

### 35.3.22 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 <a href="#">BDM mode</a> .
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

## FTMx\_CONF field descriptions (continued)

Field	Description
4–0 NUMTOF	<p>TOF Frequency</p> <p>Selects the ratio between the number of counter overflows to the number of times the TOF bit is set.</p> <p>NUMTOF = 0: The TOF bit is set for each counter overflow.</p> <p>NUMTOF = 1: The TOF bit is set for the first counter overflow but not for the next overflow.</p> <p>NUMTOF = 2: The TOF bit is set for the first counter overflow but not for the next 2 overflows.</p> <p>NUMTOF = 3: The TOF bit is set for the first counter overflow but not for the next 3 overflows.</p> <p>This pattern continues up to a maximum of 31.</p>

## 35.3.23 FTM Fault Input Polarity (FTMx\_FLTPOL)

This register defines the fault inputs polarity.

Address: Base address + 88h 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												FLT3POL	FLT2POL	FLT1POL	FLT0POL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## FTMx\_FLTPOL field descriptions

Field	Description
31–4 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
3 FLT3POL	<p>Fault Input 3 Polarity</p> <p>Defines the polarity of the fault input.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p> <p>0 The fault input polarity is active high. A 1 at the fault input indicates a fault.</p> <p>1 The fault input polarity is active low. A 0 at the fault input indicates a fault.</p>
2 FLT2POL	<p>Fault Input 2 Polarity</p> <p>Defines the polarity of the fault input.</p> <p>This field is write protected. It can be written only when MODE[WPDIS] = 1.</p>

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.

**35.3.24 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	[Shaded]															
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	HWTRIGMOD
W	[Shaded]									[Shaded]			[Shaded]		[Shaded]	E
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 HWRBUF	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 SWRBUF	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)**

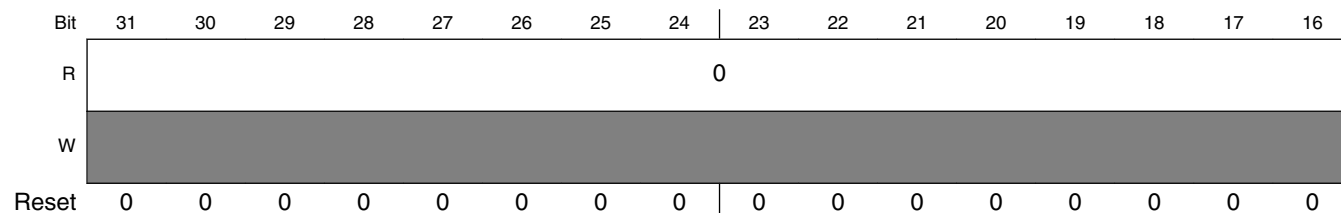
Field	Description
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SWOC	SWOCTRL Register Synchronization 0 SWOCTRL register is updated with its buffer value at all rising edges of system clock. 1 SWOCTRL register is updated with its buffer value by the 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.

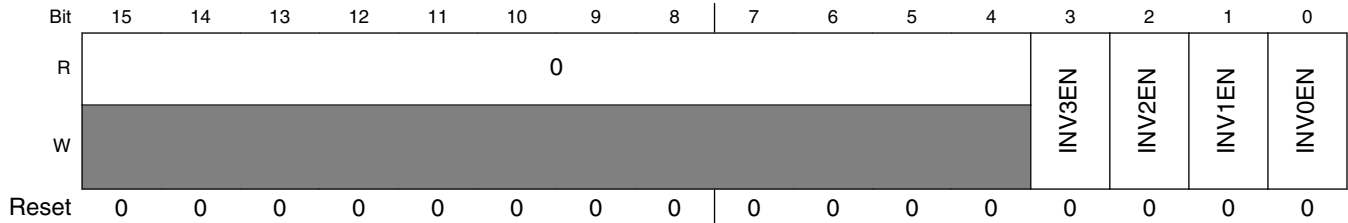
**35.3.25 FTM Inverting Control (FTMx\_INVCTRL)**

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.

### 35.3.26 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.

## Memory map and register definition

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...

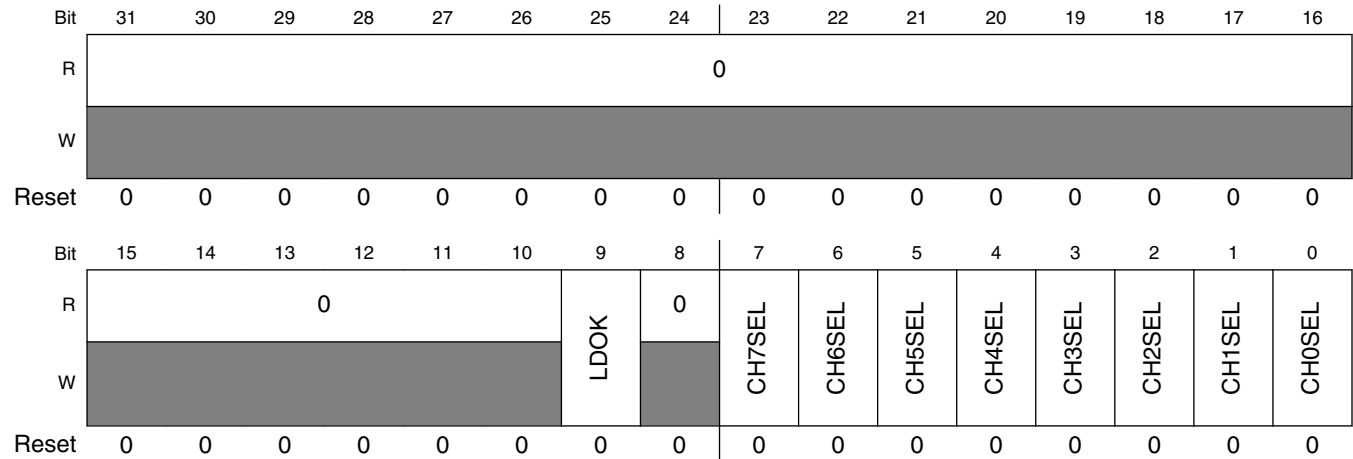
**FTMx\_SWOCTRL field descriptions (continued)**

<b>Field</b>	<b>Description</b>
	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.

### 35.3.27 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



**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)**

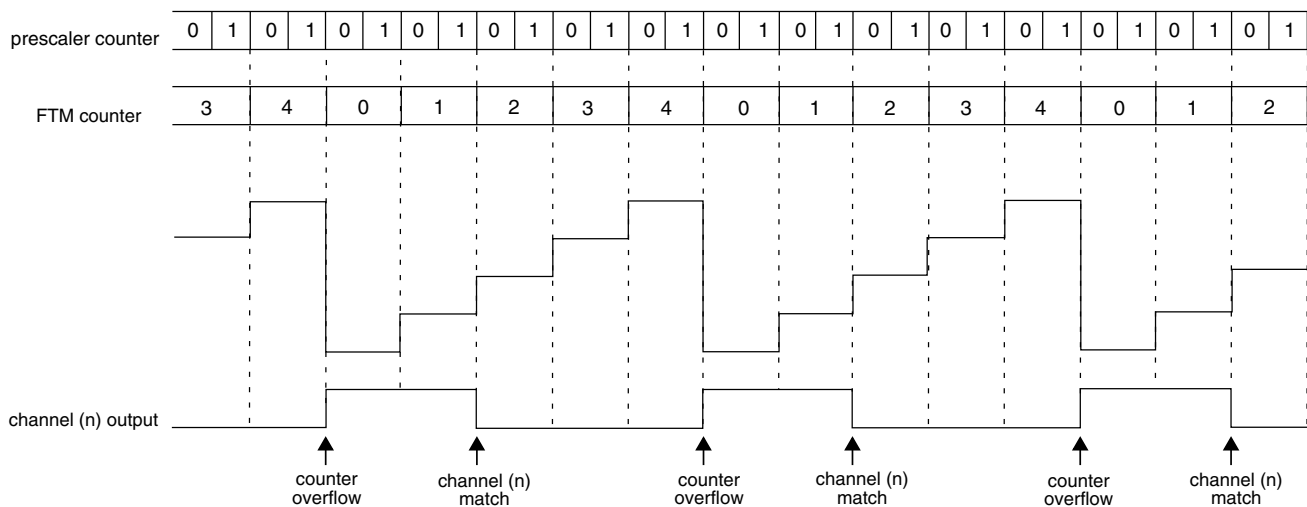
<b>Field</b>	<b>Description</b>
4 CH4SEL	Channel 4 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
3 CH3SEL	Channel 3 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
2 CH2SEL	Channel 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.

## 35.4 Functional description

The notation used in this document to represent the counters and the generation of the signals is shown in the following figure.

## Functional description

FTM counting is up.  
Channel (n) is in high-true EPWM mode.  
PS[2:0] = 001  
CNTIN = 0x0000  
MOD = 0x0004  
CnV = 0x0002



**Figure 35-166. Notation used**

## 35.4.1 Clock source

The FTM has only one clock domain: the system clock.

### 35.4.1.1 Counter clock source

The CLKS[1:0] bits in the SC register select one of three possible clock sources for the FTM counter or disable the FTM counter. After any MCU 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.

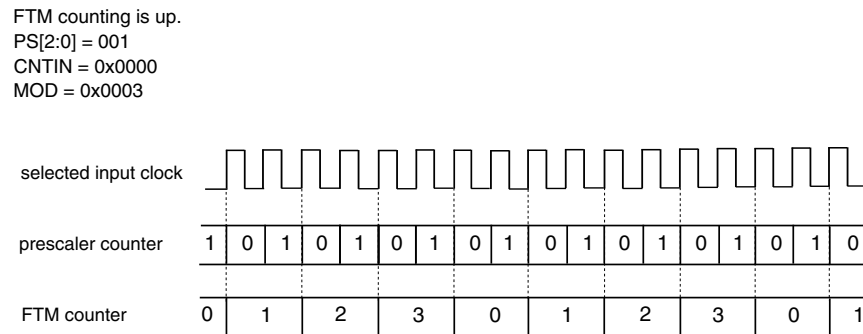
The fixed frequency clock is an alternative clock source for the FTM counter that allows the selection of a clock other than the system clock or an external clock. This clock input is defined by chip integration. Refer to the chip specific documentation for further information. Due to FTM hardware implementation 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.

## 35.4.2 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 35-167. Example of the prescaler counter**

## 35.4.3 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](#)

### 35.4.3.1 Up counting

Up counting is selected when:

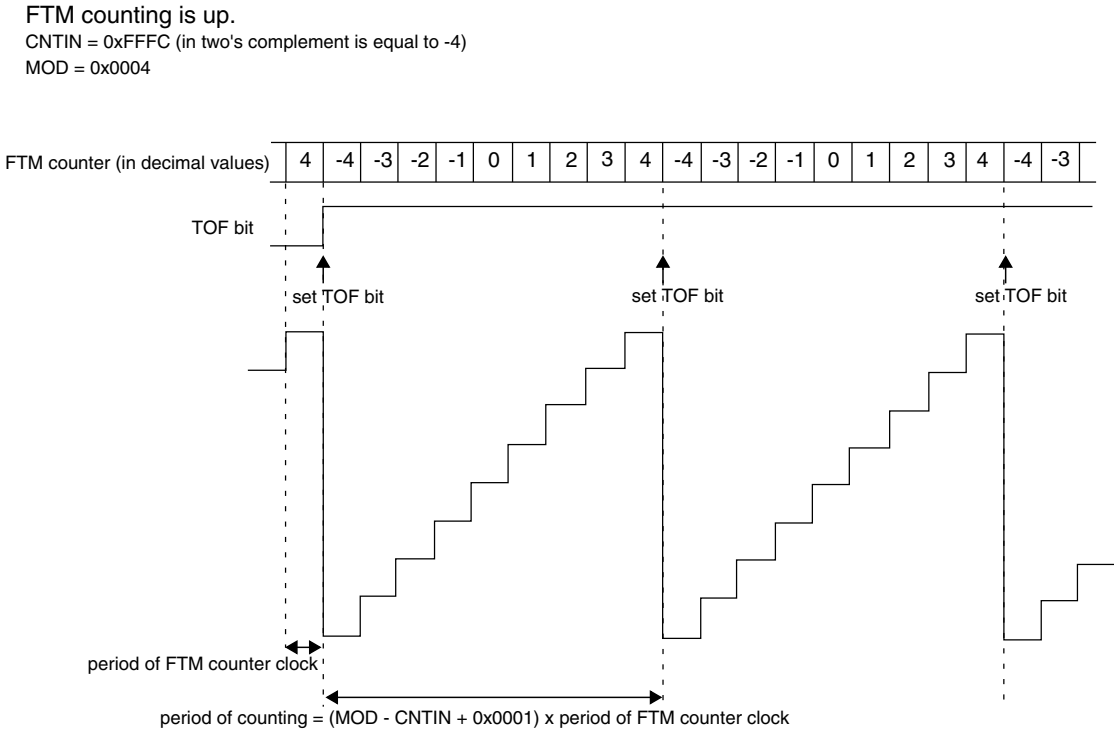
- QUADEN = 0, and
- CPWMS = 0

**Functional description**

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  $(MOD - CNTIN + 0x0001) \times$  period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to CNTIN.



**Figure 35-168. Example of FTM up and signed counting**

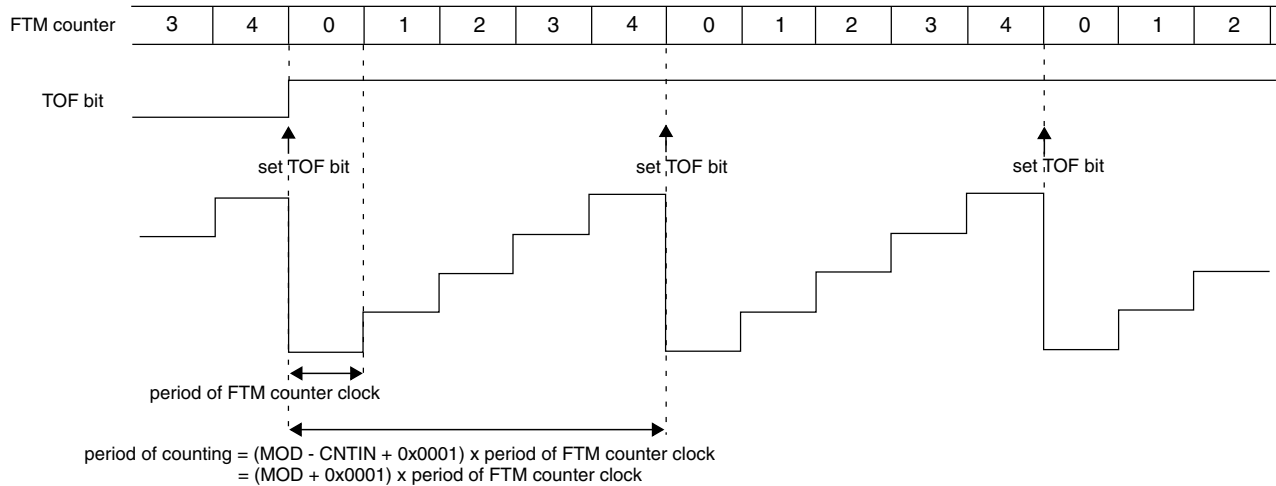
**Table 35-242. FTM counting based on CNTIN value**

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 $\neq$ 0x0000	The initial value of the FTM counter is a positive number, so the FTM counting is up and unsigned.

FTM counting is up

CNTIN = 0x0000

MOD = 0x0004

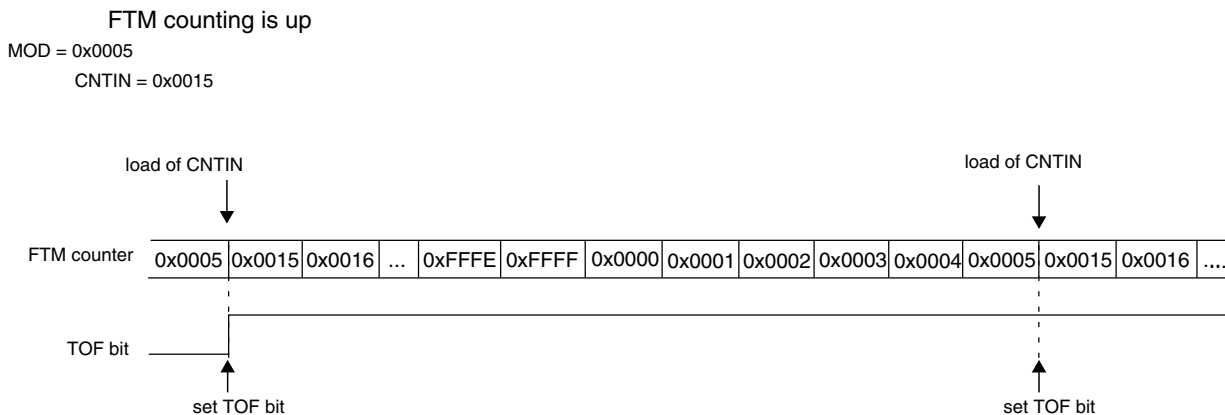


**Figure 35-169. Example of FTM up counting with CNTIN = 0x0000**

### 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.

## Functional description



**Figure 35-170. Example of up counting when the value of CNTIN is greater than the value of MOD**

### 35.4.3.2 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  $2 \times (\text{MOD} - \text{CNTIN}) \times \text{period of the FTM counter clock}$ .

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.

FTM counting is up-down  
 CNTIN = 0x0000  
 MOD = 0x0004

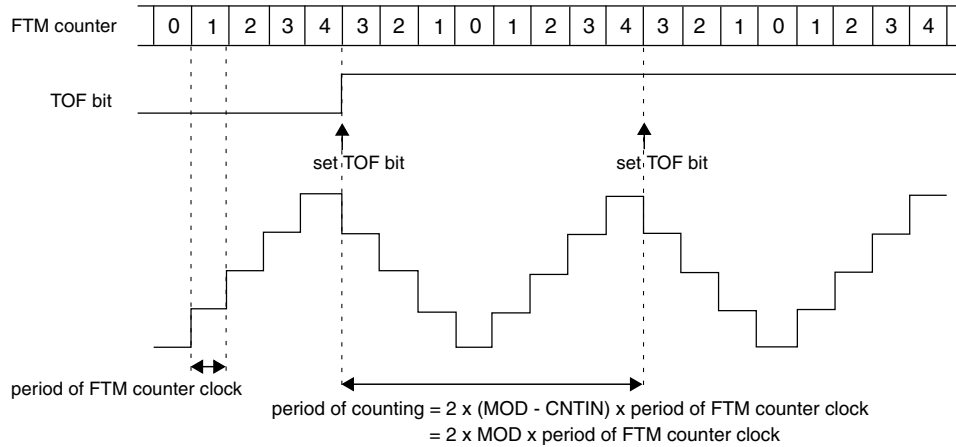


Figure 35-171. Example of up-down counting when CNTIN = 0x0000

**Note**

When CNTIN is different from zero in the up-down counting, a valid CPWM signal is generated:

- if  $C_nV > \text{CNTIN}$ , or
- if  $C_nV = 0$  or if  $C_nV[15] = 1$ . In this case, 0% CPWM is generated.

**35.4.3.3 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.

FTMEN = 0  
 MOD = 0x0000

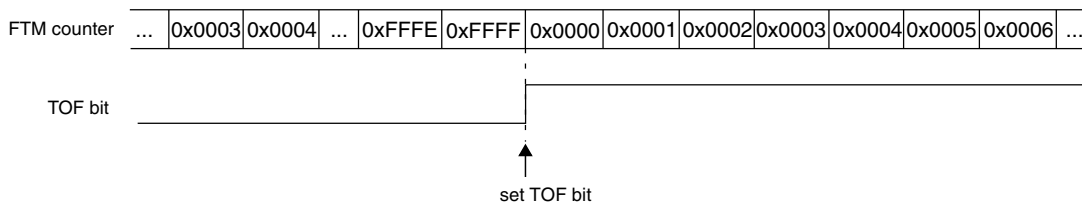


Figure 35-172. Example when the FTM counter is free running

The FTM counter is also a free running counter when:

## Functional description

- FTMEN = 1
- QUADEN = 0
- CPWMS = 0
- CNTIN = 0x0000, and
- MOD = 0xFFFF

### 35.4.3.4 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](#)).

### 35.4.3.5 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.

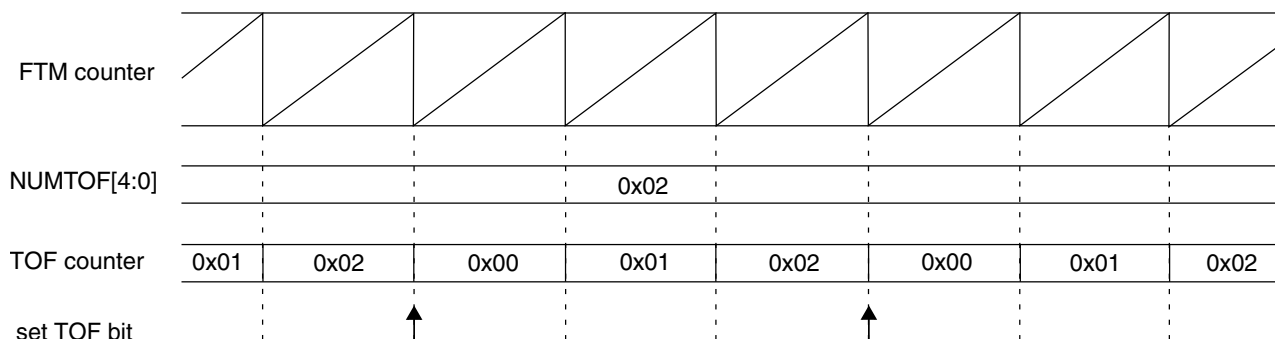


Figure 35-173. Periodic TOF when NUMTOF = 0x02

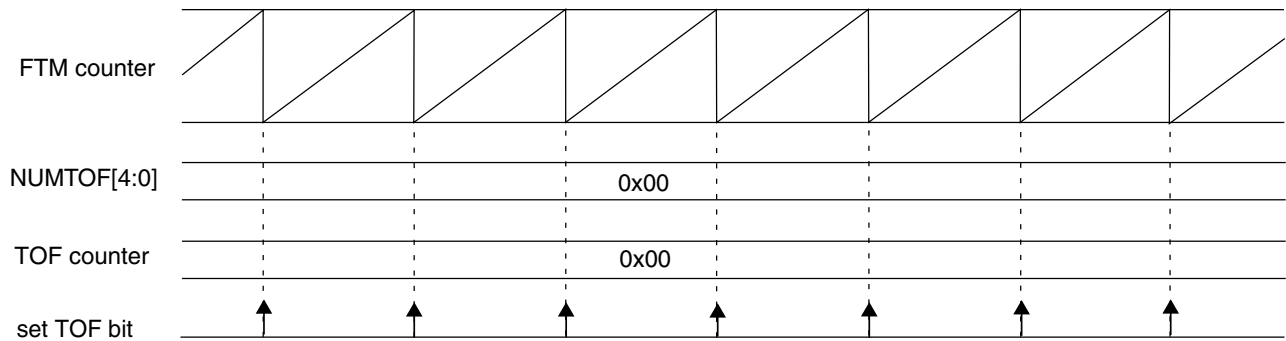


Figure 35-174. Periodic TOF when NUMTOF = 0x00

### 35.4.4 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

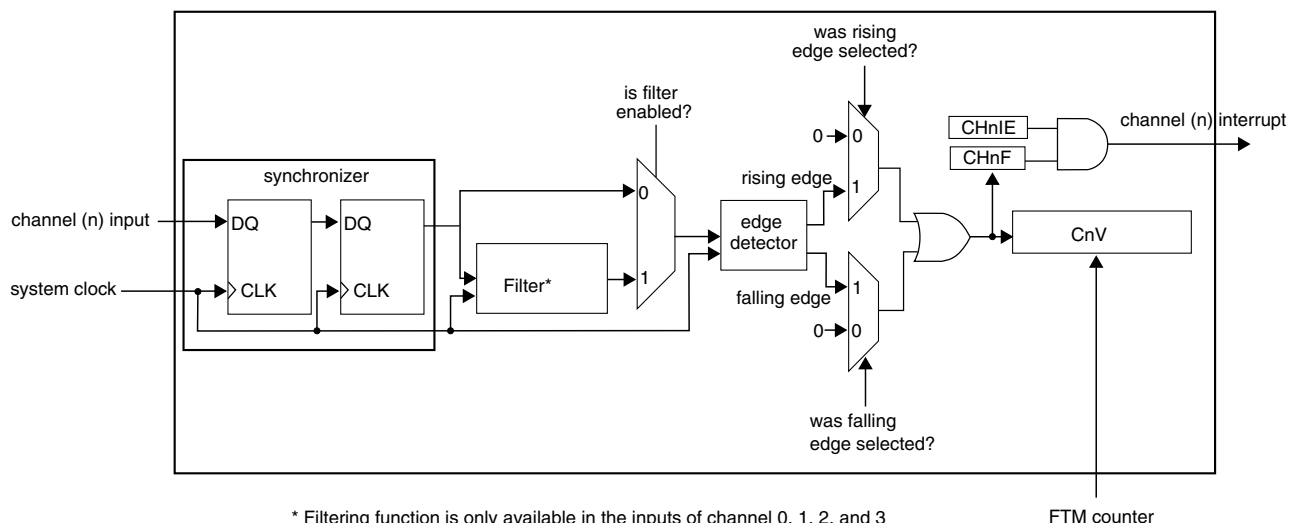
When a selected edge occurs on the channel input, the current value of the FTM counter is captured into the CnV register, at the same time the CHnF bit is set and the channel interrupt is generated if enabled by CHnIE = 1. See the following figure.

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.

## Functional description



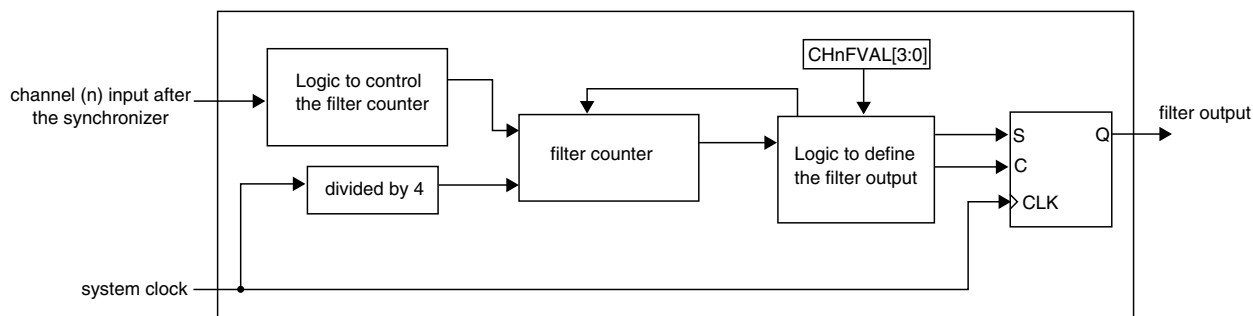
**Figure 35-175. Input Capture mode**

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.

### 35.4.4.1 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 35-176. Channel input filter**

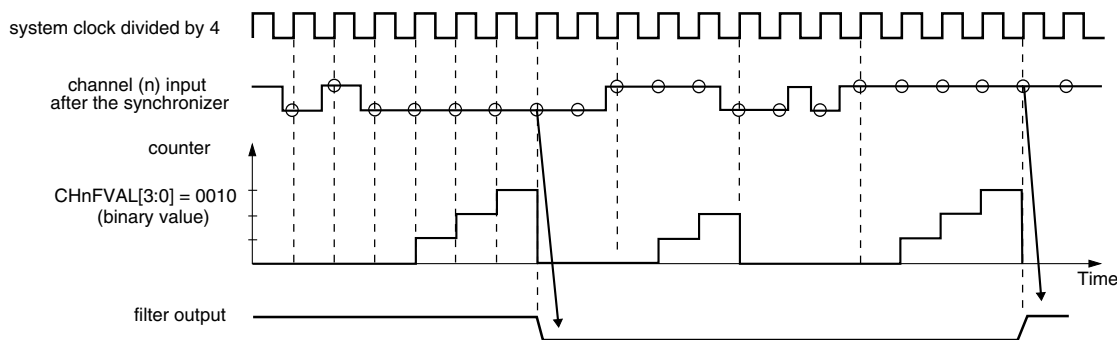
When there is a state change in the input signal, the counter is reset and starts counting up. As long as the new state is stable on the input, the counter continues to increment. When the counter is equal to CHnFVAL[3:0], the state change of the input signal is validated. It is then transmitted as a 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  $CHnFVAL[3:0]$  ( $\times 4$  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 ( $CHnFVAL[3:0] \neq 0000$ ), then the input signal is delayed by the minimum pulse width ( $CHnFVAL[3:0] \times 4$  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  $(4 + 4 \times CHnFVAL[3:0])$  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.



**Figure 35-177. Channel input filter example**

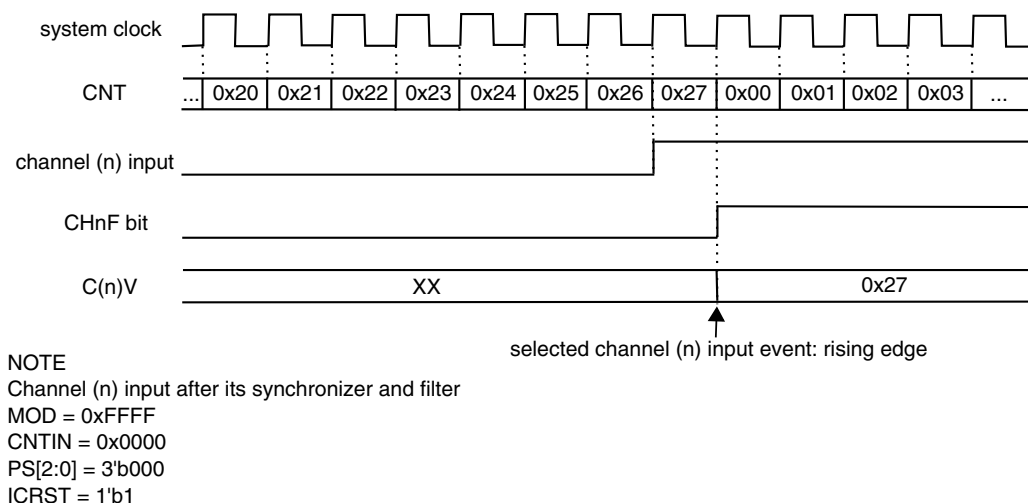
#### 35.4.4.2 FTM Counter Reset in Input Capture Mode

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  $ICRST = 1$ .

## Functional description



**Figure 35-178. 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.

## 35.4.5 Output Compare mode

The Output Compare mode is selected when:

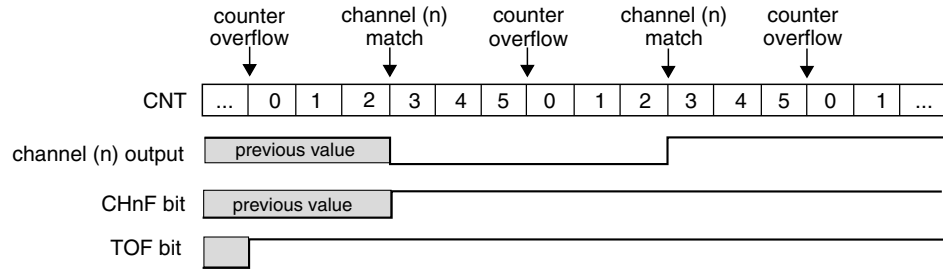
- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB:MSnA = 0:1

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.

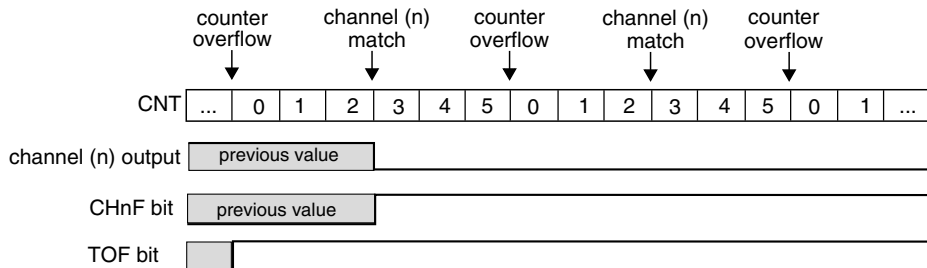
The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV).

MOD = 0x0005  
CnV = 0x0003



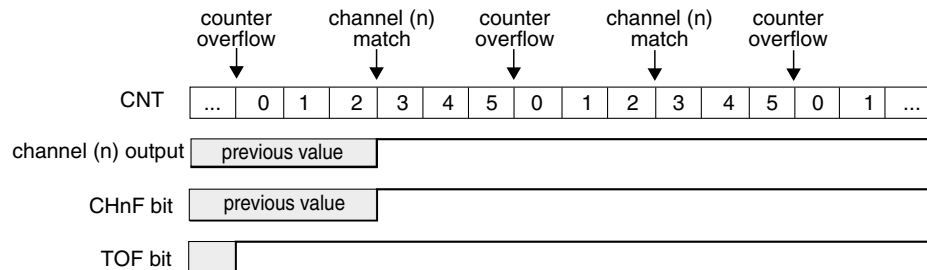
**Figure 35-179. Example of the Output Compare mode when the match toggles the channel output**

MOD = 0x0005  
CnV = 0x0003



**Figure 35-180. Example of the Output Compare mode when the match clears the channel output**

MOD = 0x0005  
CnV = 0x0003



**Figure 35-181. 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.

### 35.4.6 Edge-Aligned PWM (EPWM) mode

The Edge-Aligned mode is selected when:

- QUADEN = 0

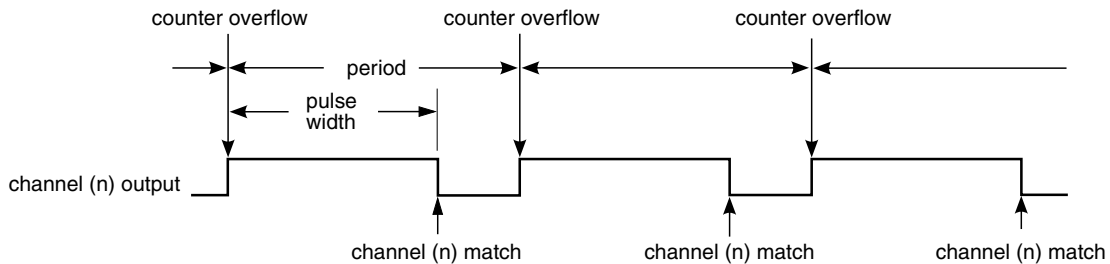
## Functional description

- 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.

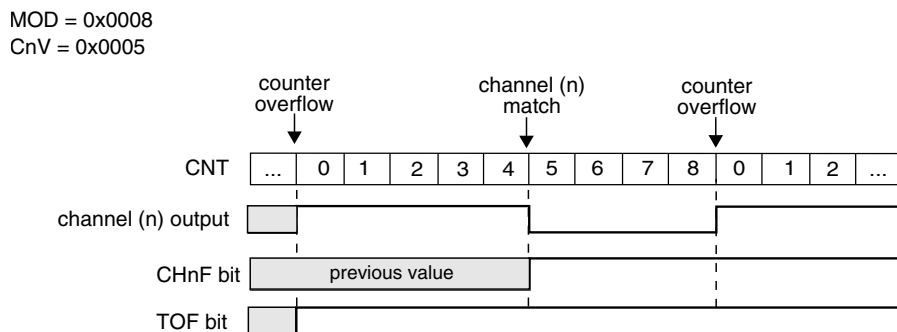
This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an FTM.



**Figure 35-182. EPWM period and pulse width with ELSnB:ELSnA = 1:0**

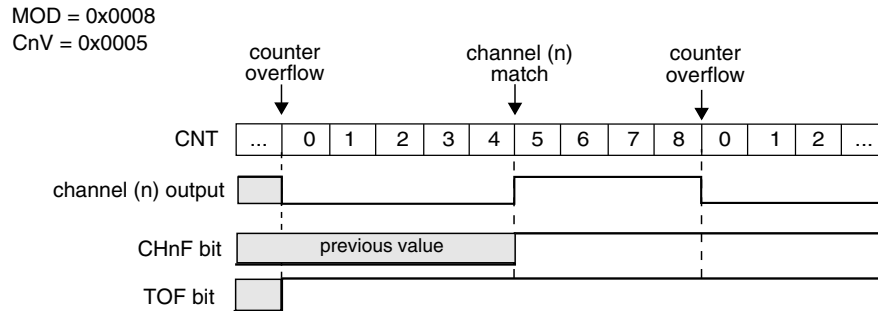
If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not controlled by FTM.

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 35-183. 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 35-184. 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 even when there is the channel (n) match. 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  $CNTIN < CnV \leq MOD$ ,
- 100% EPWM signal when  $CNTIN > CnV$  or  $CnV > MOD$ .

## 35.4.7 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  $2 \times (CnV - CNTIN)$  and the period is determined by  $2 \times (MOD - CNTIN)$ . See the following figure. MOD must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

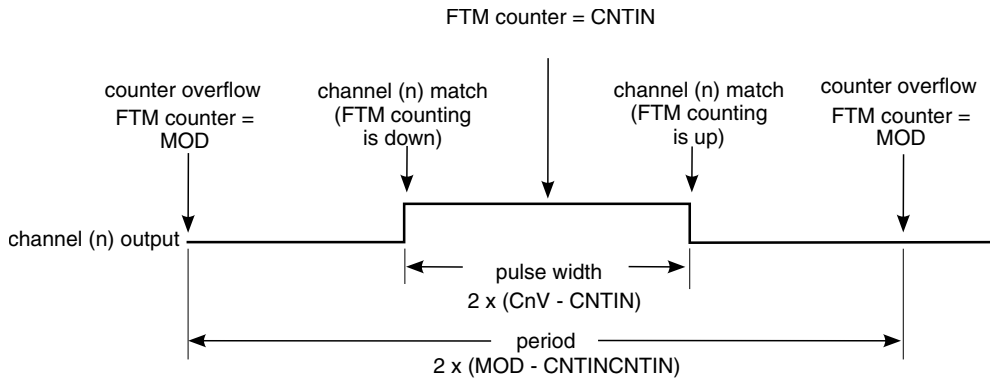
In the CPWM mode, the FTM counter counts up until it reaches MOD and then counts down until it reaches CNTIN.

## Functional description

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.

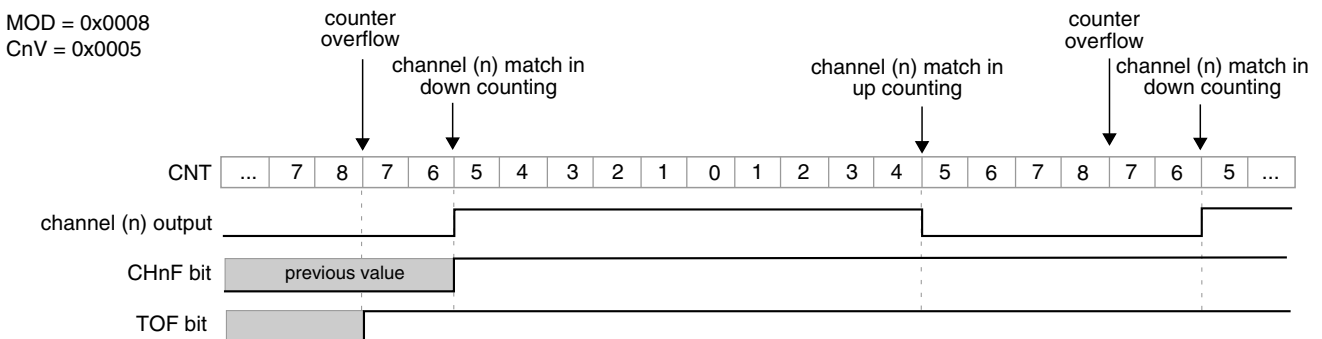
The other channel modes are not compatible with the up-down counter (CPWMS = 1). Therefore, all FTM channels must be used in CPWM mode when (CPWMS = 1).



**Figure 35-185. CPWM period and pulse width with ELSnB:ELSnA = 1:0**

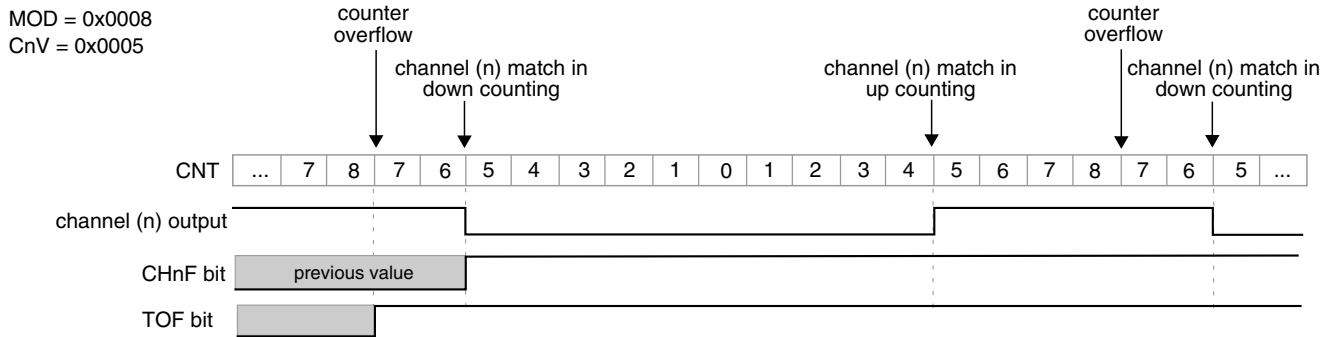
If (ELSnB:ELSnA = 0:0) when the FTM counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by FTM.

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 35-186. 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 35-187. 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 ( $CnV[15] = 0$ ), ( $CnV \geq MOD$ ), and ( $MOD \neq 0x0000$ ), 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.

### 35.4.8 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  $(IC(n+1)V - C(n)V)$ .

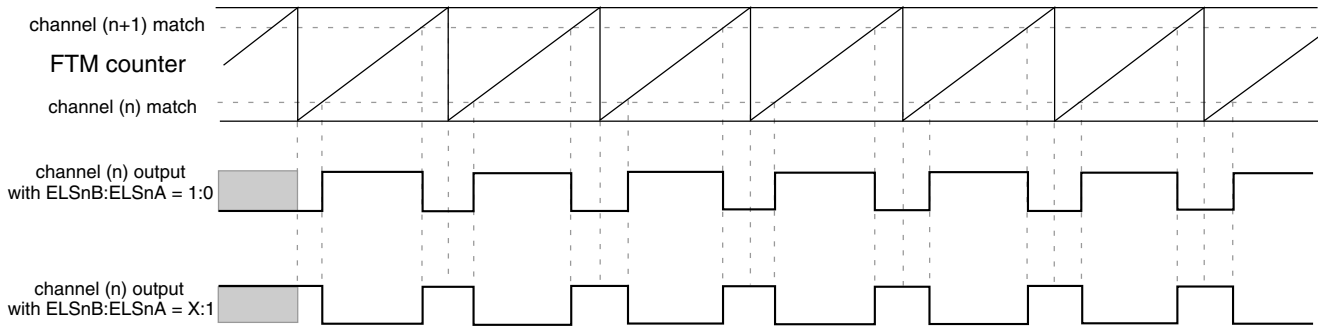
The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter =  $C(n)V$ ). The CH(n+1)F bit is set and the channel (n+1) interrupt is generated, if CH(n+1)IE = 1, at the channel (n+1) match (FTM counter =  $C(n+1)V$ ).

## Functional description

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.

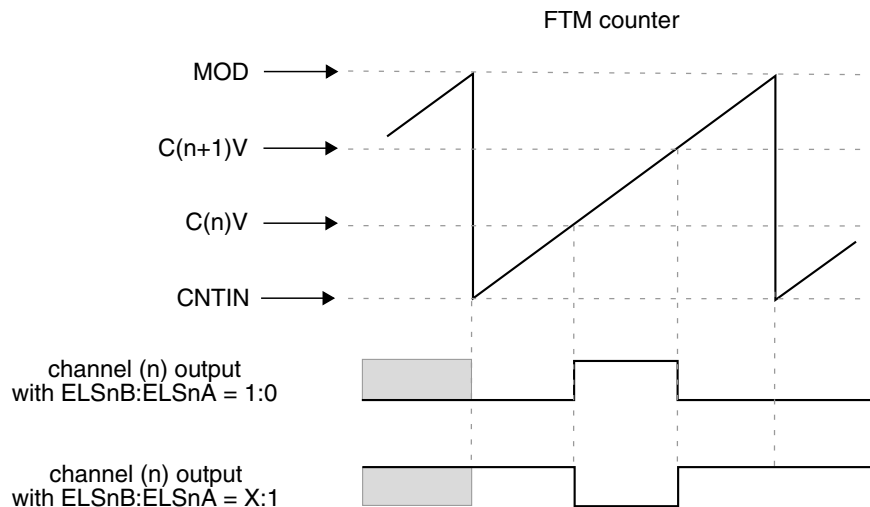
If (ELSnB:ELSnA = X:1), then the channel (n) output is forced high at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n+1)V). It is forced low at the channel (n) match (FTM counter = C(n)V). See the following figure.

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 (ELSnB:ELSnA = 0:0) then the channel (n+1) output is not controlled by FTM.



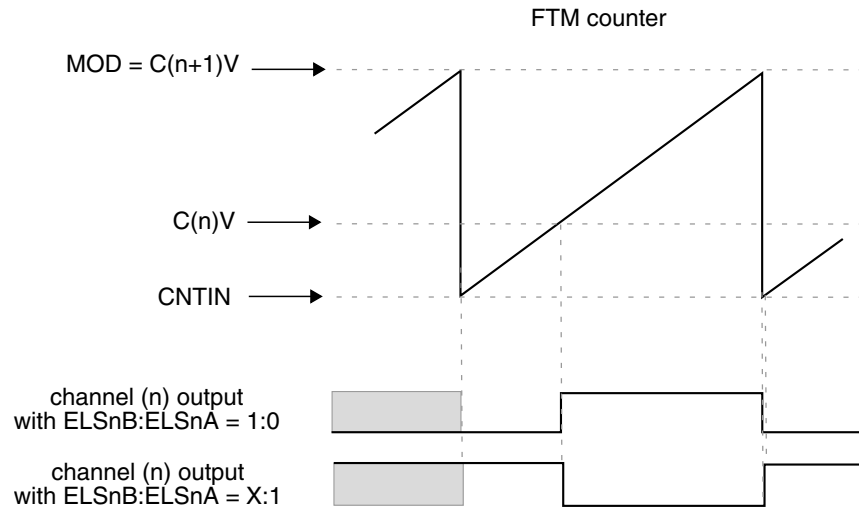
**Figure 35-188. Combine mode**

The following figures illustrate the PWM signals generation using Combine mode.

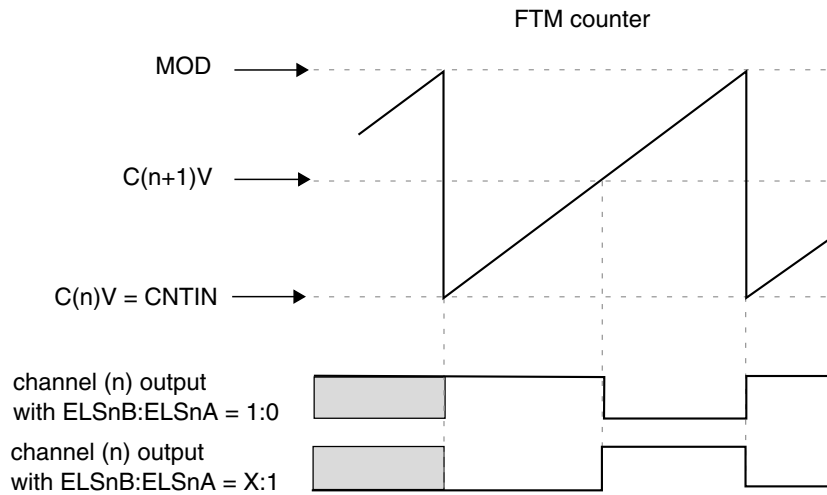


**Figure 35-189. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V < C(n+1)V)**

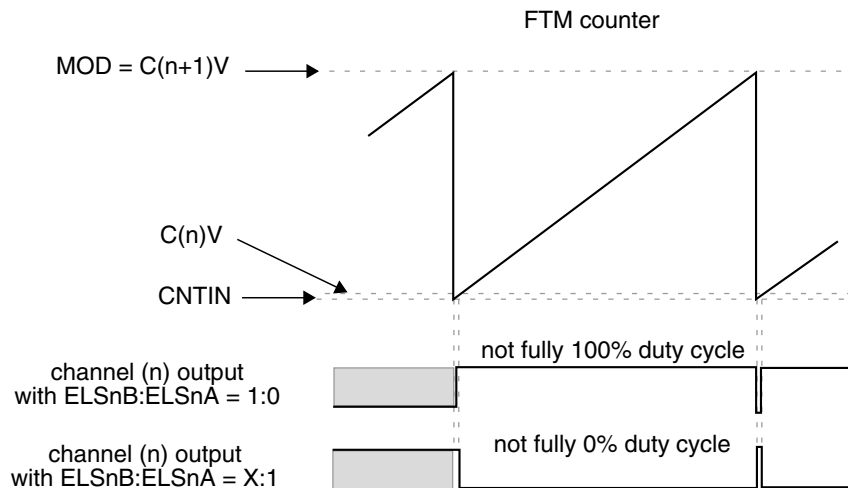




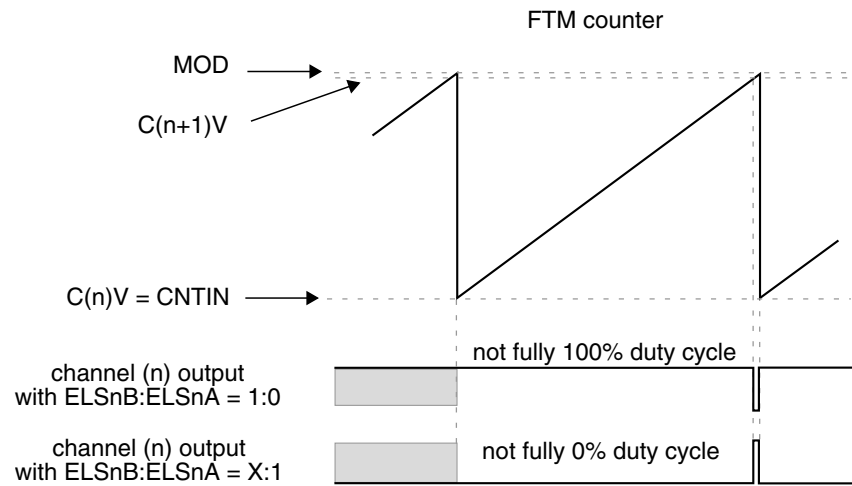
**Figure 35-190. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n+1)V = MOD)**



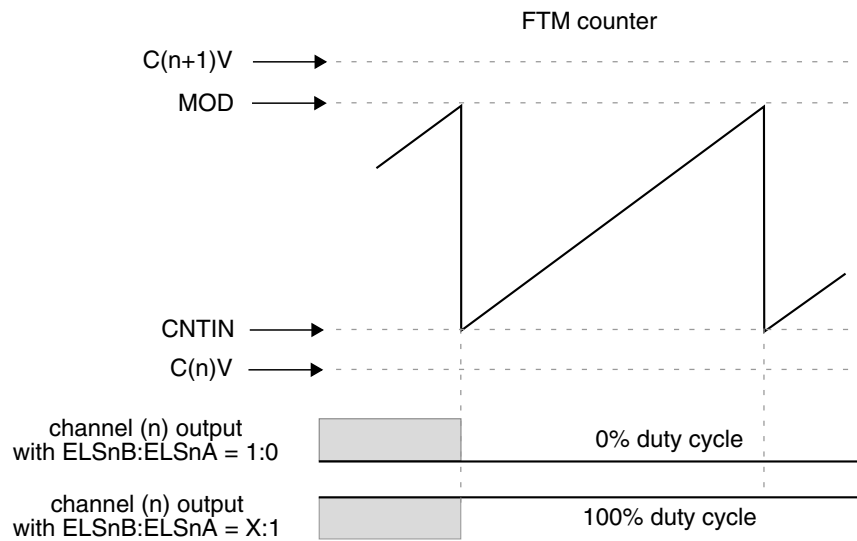
**Figure 35-191. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD)**



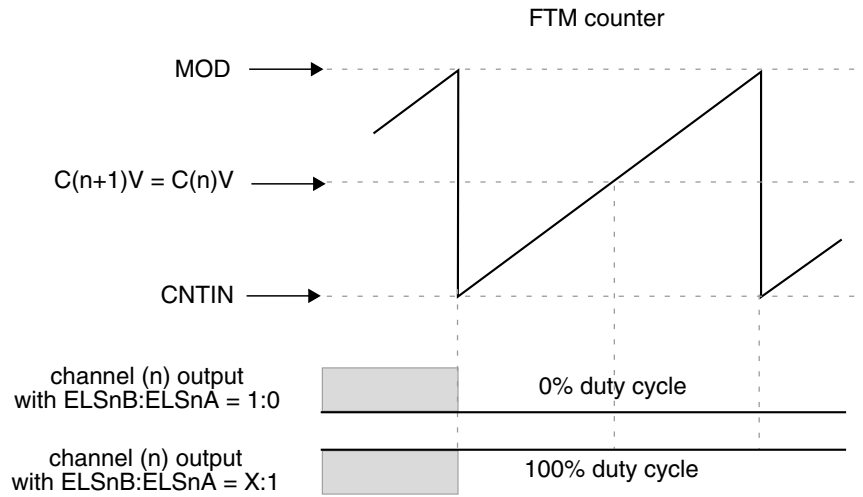
**Figure 35-192. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n)V is Almost Equal to CNTIN) and (C(n+1)V = MOD)**



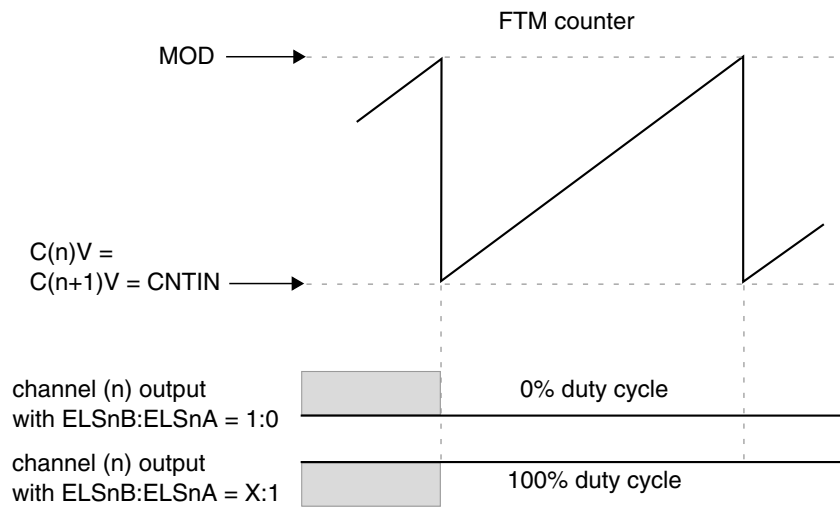
**Figure 35-193. Channel (n) output if  $(C(n)V = CNTIN)$  and  $(CNTIN < C(n+1)V < MOD)$  and  $(C(n+1)V$  is Almost Equal to  $MOD)$**



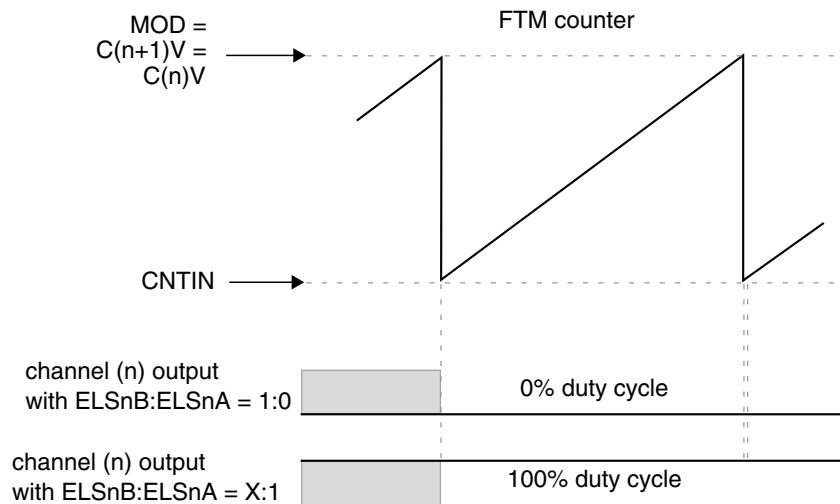
**Figure 35-194. Channel (n) output if  $C(n)V$  and  $C(n+1)V$  are not between  $CNTIN$  and  $MOD$**



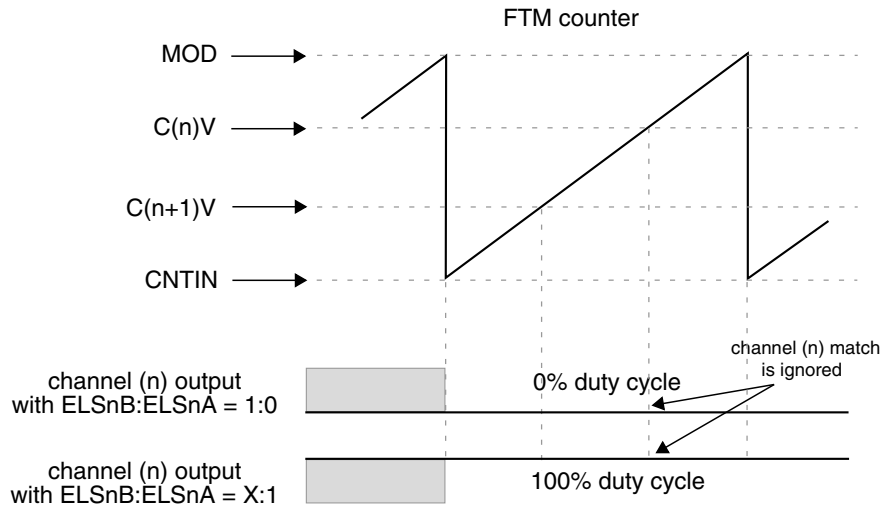
**Figure 35-195. Channel (n) output if  $(CNTIN < C(n)V < MOD)$  and  $(CNTIN < C(n+1)V < MOD)$  and  $(C(n)V = C(n+1)V)$**



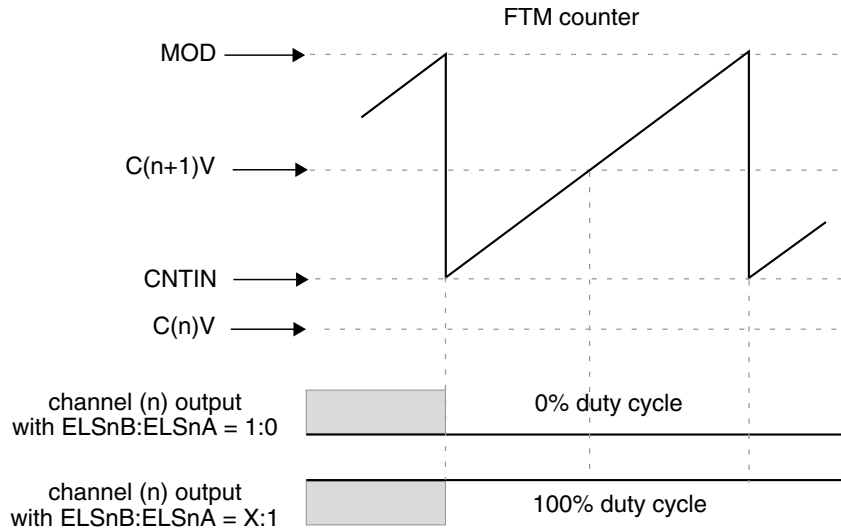
**Figure 35-196. Channel (n) output if  $(C(n)V = C(n+1)V = CNTIN)$**



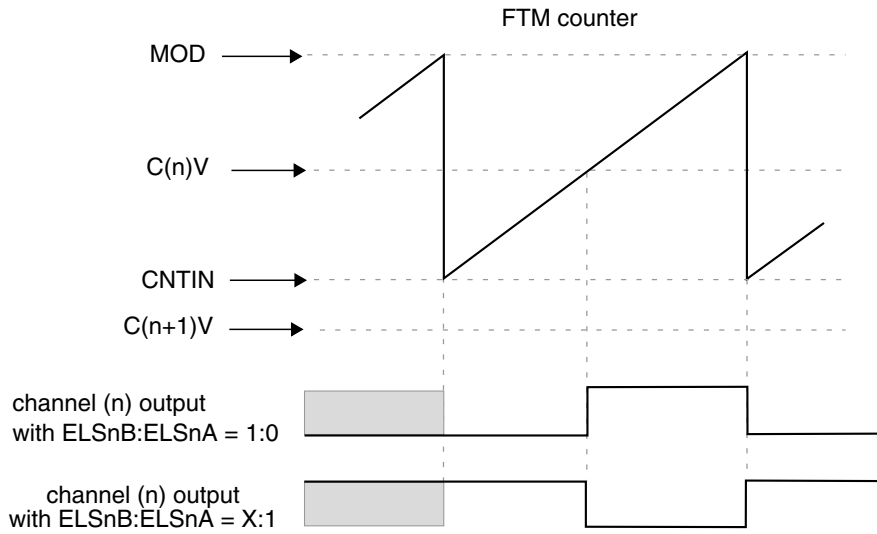
**Figure 35-197. Channel (n) output if  $(C(n)V = C(n+1)V = MOD)$**



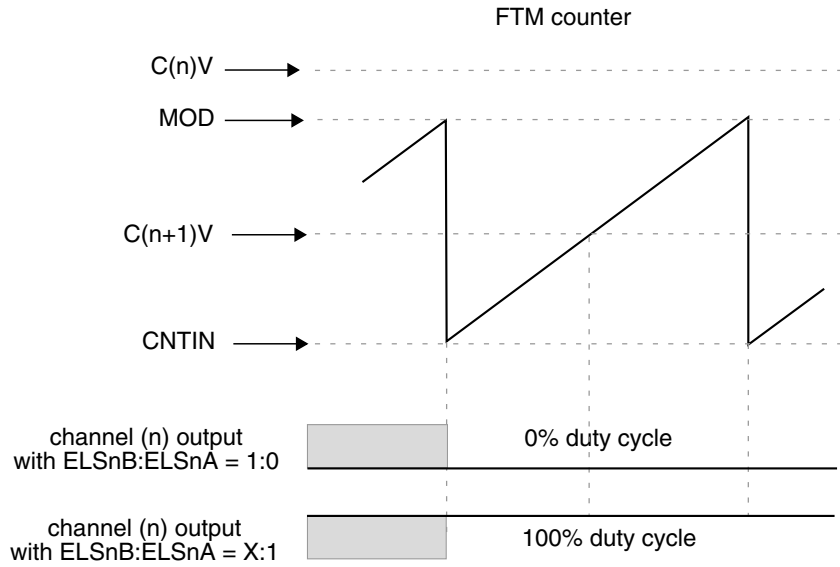
**Figure 35-198. Channel (n) output if  $(CNTIN < C(n)V < MOD)$  and  $(CNTIN < C(n+1)V < MOD)$  and  $(C(n)V > C(n+1)V)$**



**Figure 35-199. Channel (n) output if  $(C(n)V < CNTIN)$  and  $(CNTIN < C(n+1)V < MOD)$**



**Figure 35-200. Channel (n) output if  $(C(n+1)V < CNTIN)$  and  $(CNTIN < C(n)V < MOD)$**



**Figure 35-201. Channel (n) output if  $(C(n)V > MOD)$  and  $(CNTIN < C(n+1)V < MOD)$**

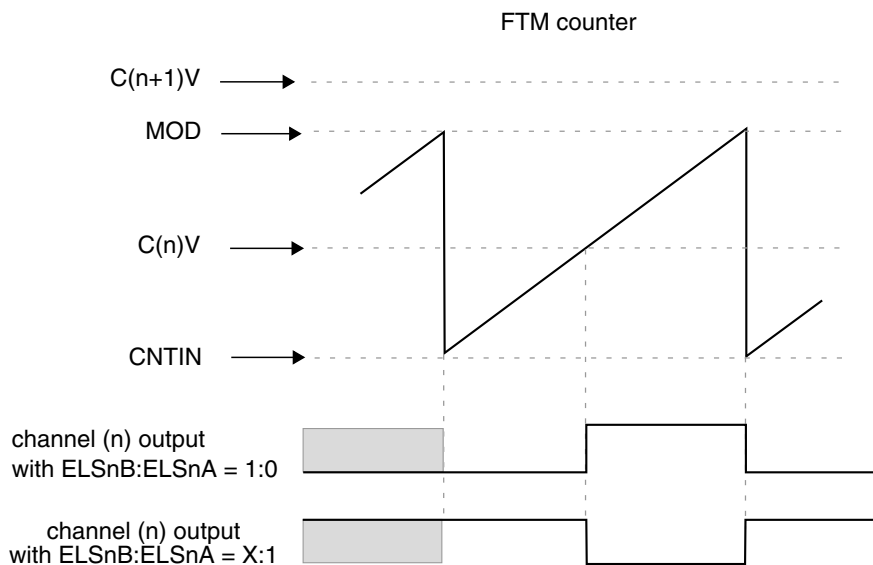


Figure 35-202. Channel (n) output if  $(C(n+1)V > MOD)$  and  $(CNTIN < C(n)V < MOD)$

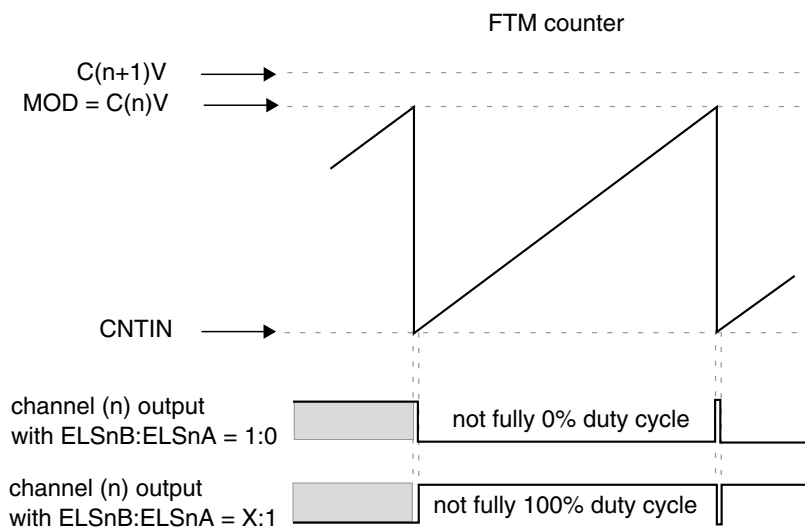


Figure 35-203. Channel (n) output if  $(C(n+1)V > MOD)$  and  $(CNTIN < C(n)V = MOD)$

### 35.4.8.1 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.

### 35.4.9 Complementary mode

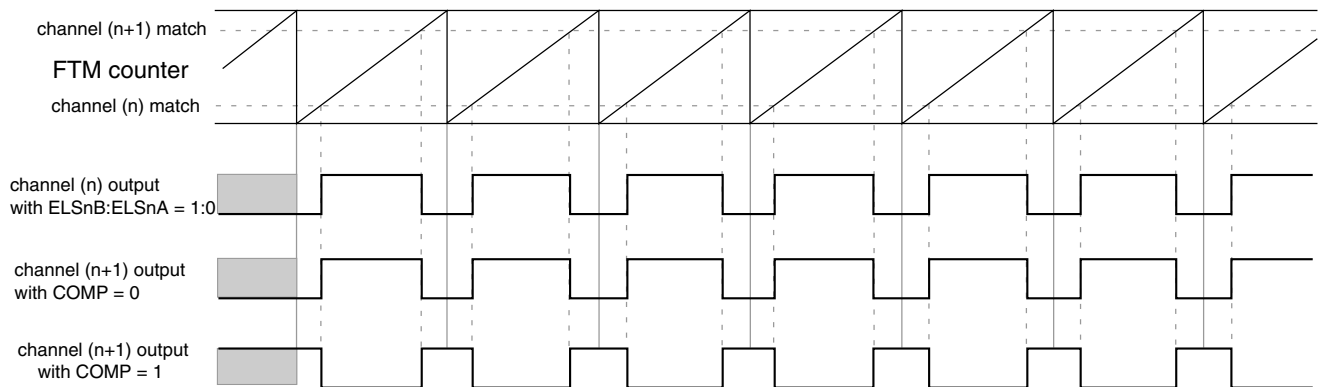
The Complementary mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 1

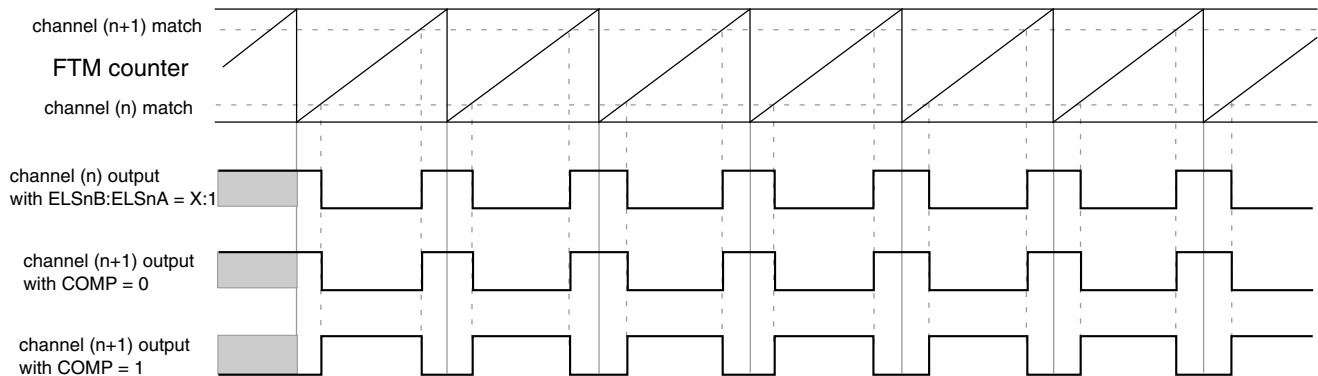
In Complementary mode, the channel (n+1) output is the inverse of the channel (n) output.

So, the channel (n+1) output is the same as the channel (n) output when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 0



**Figure 35-204. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = 1:0)**



**Figure 35-205. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = X:1)**

### NOTE

The complementary mode is not available in Output Compare mode.

## 35.4.10 Registers updated from write buffers

### 35.4.10.1 CNTIN register update

The following table describes when CNTIN register is updated:

**Table 35-243. CNTIN register update**

When	Then CNTIN register is updated
CLKS[1:0] = 0:0	When CNTIN register is written, independent of FTMEN bit.
<ul style="list-style-type: none"> <li>• FTMEN = 0, or</li> <li>• CNTINC = 0</li> </ul>	At the next system clock after CNTIN was written.
<ul style="list-style-type: none"> <li>• FTMEN = 1,</li> <li>• SYNCMODE = 1, and</li> <li>• CNTINC = 1</li> </ul>	By the <a href="#">CNTIN register synchronization</a> .

### 35.4.10.2 MOD register update

The following table describes when MOD register is updated:

**Table 35-244. MOD register update**

When	Then MOD register is updated
CLKS[1:0] = 0:0	When MOD register is written, independent of FTMEN bit.
<ul style="list-style-type: none"> <li>• CLKS[1:0] ≠ 0:0, and</li> <li>• FTMEN = 0</li> </ul>	According to the CPWMS bit, that is: <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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).</li> </ul>
<ul style="list-style-type: none"> <li>• CLKS[1:0] ≠ 0:0, and</li> <li>• FTMEN = 1</li> </ul>	By the <a href="#">MOD register synchronization</a> .

### 35.4.10.3 CnV register update

The following table describes when CnV register is updated:

**Table 35-245. CnV register update**

When	Then CnV register is updated
CLKS[1:0] = 0:0	When CnV register is written, independent of FTMEN bit.

*Table continues on the next page...*



Table 35-245. CnV register update (continued)

When	Then CnV register is updated
<ul style="list-style-type: none"> <li>• CLKS[1:0] <math>\neq</math> 0:0, and</li> <li>• FTMEN = 0</li> </ul>	<p>According to the selected mode, that is:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• 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).</li> </ul>
<ul style="list-style-type: none"> <li>• CLKS[1:0] <math>\neq</math> 0:0, and</li> <li>• FTMEN = 1</li> </ul>	<p>According to the selected mode, that is:</p> <ul style="list-style-type: none"> <li>• 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 <a href="#">C(n)V and C(n+1)V register synchronization</a>.</li> <li>• If the selected mode is not output compare and (SYNCEN = 1) then CnV register is updated by the <a href="#">C(n)V and C(n+1)V register synchronization</a>.</li> </ul>

### 35.4.11 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.

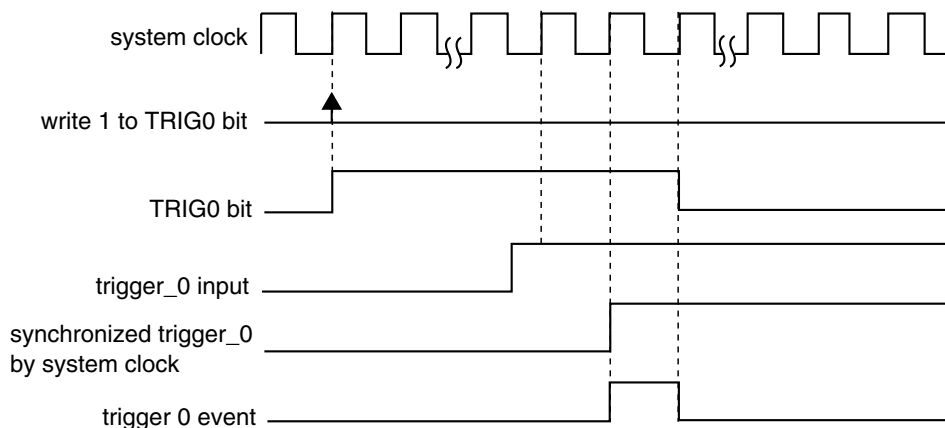
#### 35.4.11.1 Hardware trigger

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.

## Functional description

If ( $\text{HWTRIGMODE} = 0$ ) then the  $\text{TRIGn}$  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,  $\text{TRIG0} = 1$  and  $\text{TRIG1} = 1$ ) and only trigger 1 event occurs, then only  $\text{TRIG1}$  bit is cleared. If a trigger n event occurs together with a write setting  $\text{TRIGn}$  bit, then the synchronization is initiated, but  $\text{TRIGn}$  bit remains set due to the write operation.



Note  
All hardware trigger inputs have the same behavior.

**Figure 35-206. Hardware trigger event with  $\text{HWTRIGMODE} = 0$**

If  $\text{HWTRIGMODE} = 1$ , then the  $\text{TRIGn}$  bit is only cleared when 0 is written to it.

### NOTE

The  $\text{HWTRIGMODE}$  bit must be 1 only with enhanced PWM synchronization ( $\text{SYNCMODE} = 1$ ).

## 35.4.11.2 Software trigger

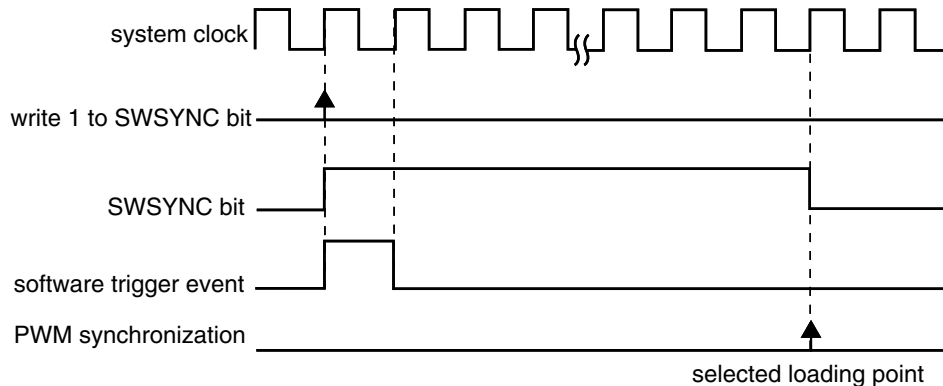
A software trigger event occurs when 1 is written to the  $\text{SYNC}[\text{SWSYNC}]$  bit. The  $\text{SWSYNC}$  bit is cleared when 0 is written to it or when the PWM synchronization, initiated by the software event, is completed.

If another software trigger event occurs (by writing another 1 to the  $\text{SWSYNC}$  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  $\text{SWSYNC}$  bit remains equal to 1.

If  $\text{SYNCMODE} = 0$  then the  $\text{SWSYNC}$  bit is also cleared by FTM according to  $\text{PWMSYNC}$  and  $\text{REINIT}$  bits. In this case if ( $\text{PWMSYNC} = 1$ ) or ( $\text{PWMSYNC} = 0$  and  $\text{REINIT} = 0$ ) then  $\text{SWSYNC}$  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 35-207. Software trigger event**

### 35.4.11.3 Boundary cycle and loading points

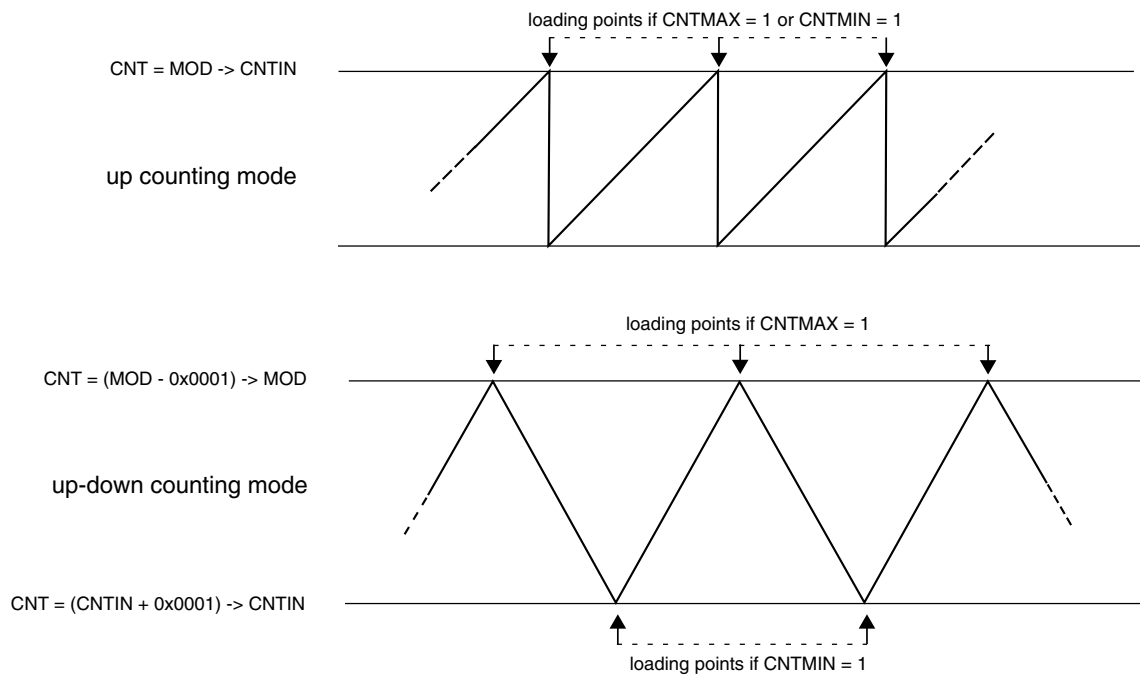
The boundary cycle definition is important for the loading points for the registers MOD, CNTIN, and C(n)V.

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.

## Functional description



**Figure 35-208. Boundary cycles and loading points**

### 35.4.11.4 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:

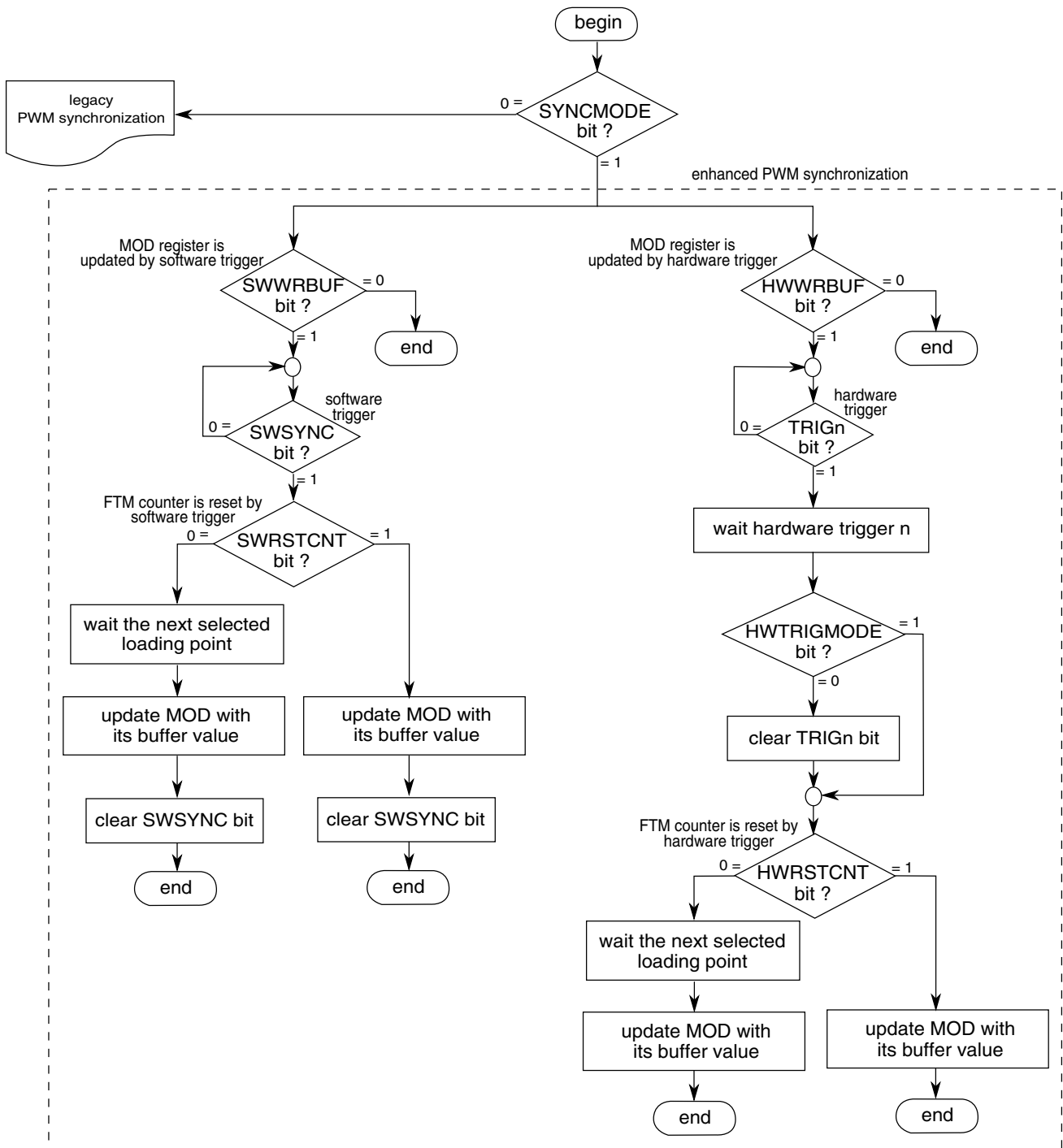


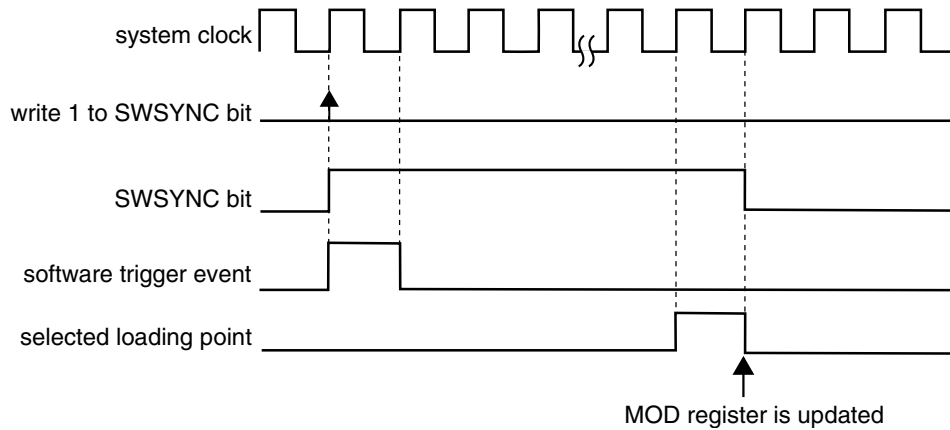
Figure 35-209. MOD register synchronization flowchart

In the case of legacy PWM synchronization, the MOD register synchronization depends on PWMSYNC and REINIT bits according to the following description.

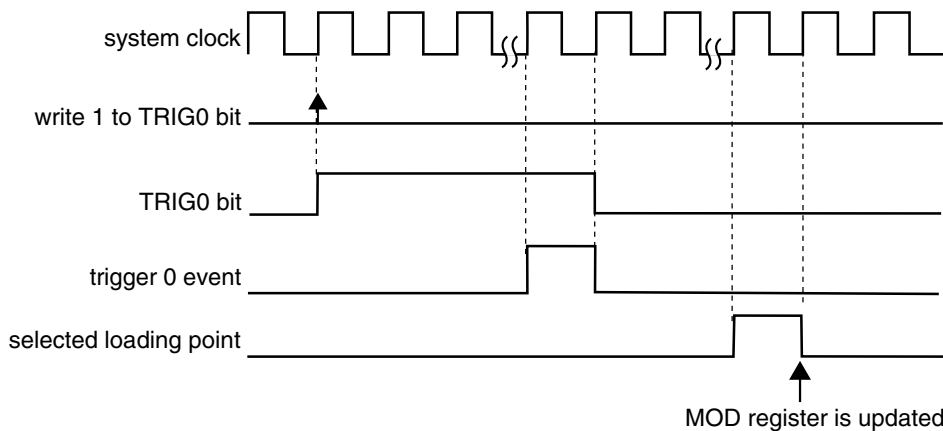
If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 0), then this synchronization is made on the next selected loading point after an enabled trigger event takes place. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected

## Functional description

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.

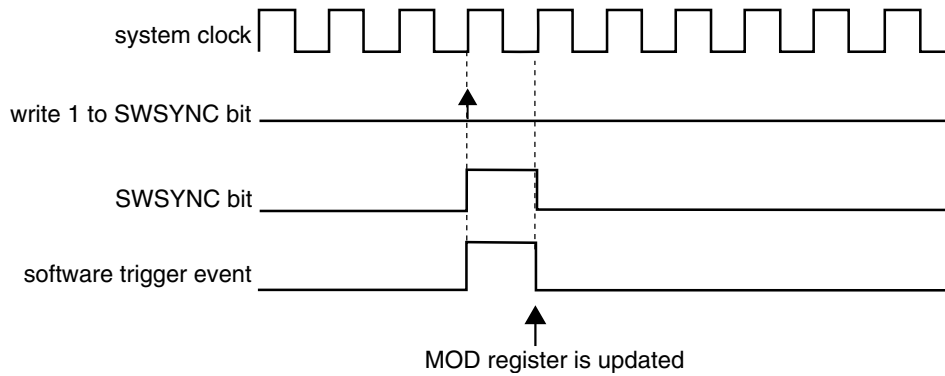


**Figure 35-210. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 0), and software trigger was used**

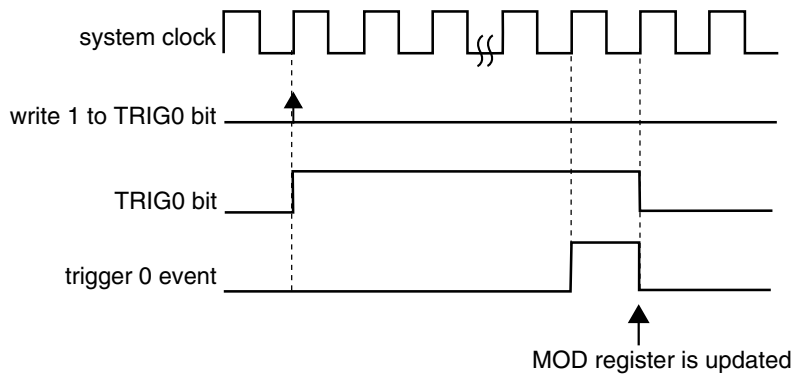


**Figure 35-211. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 0), and a hardware trigger was used**

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.

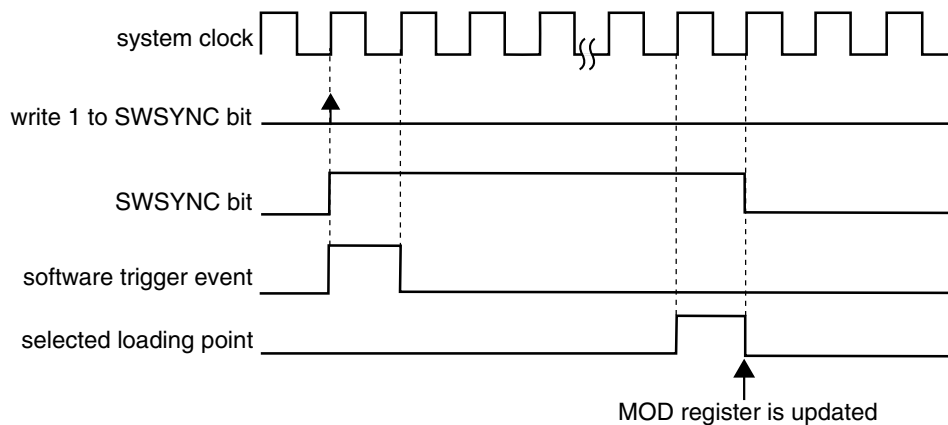


**Figure 35-212. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 1), and software trigger was used**



**Figure 35-213. 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 35-214. MOD synchronization with (SYNCMODE = 0) and (PWMSYNC = 1)**

### 35.4.11.5 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](#).

### 35.4.11.6 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).

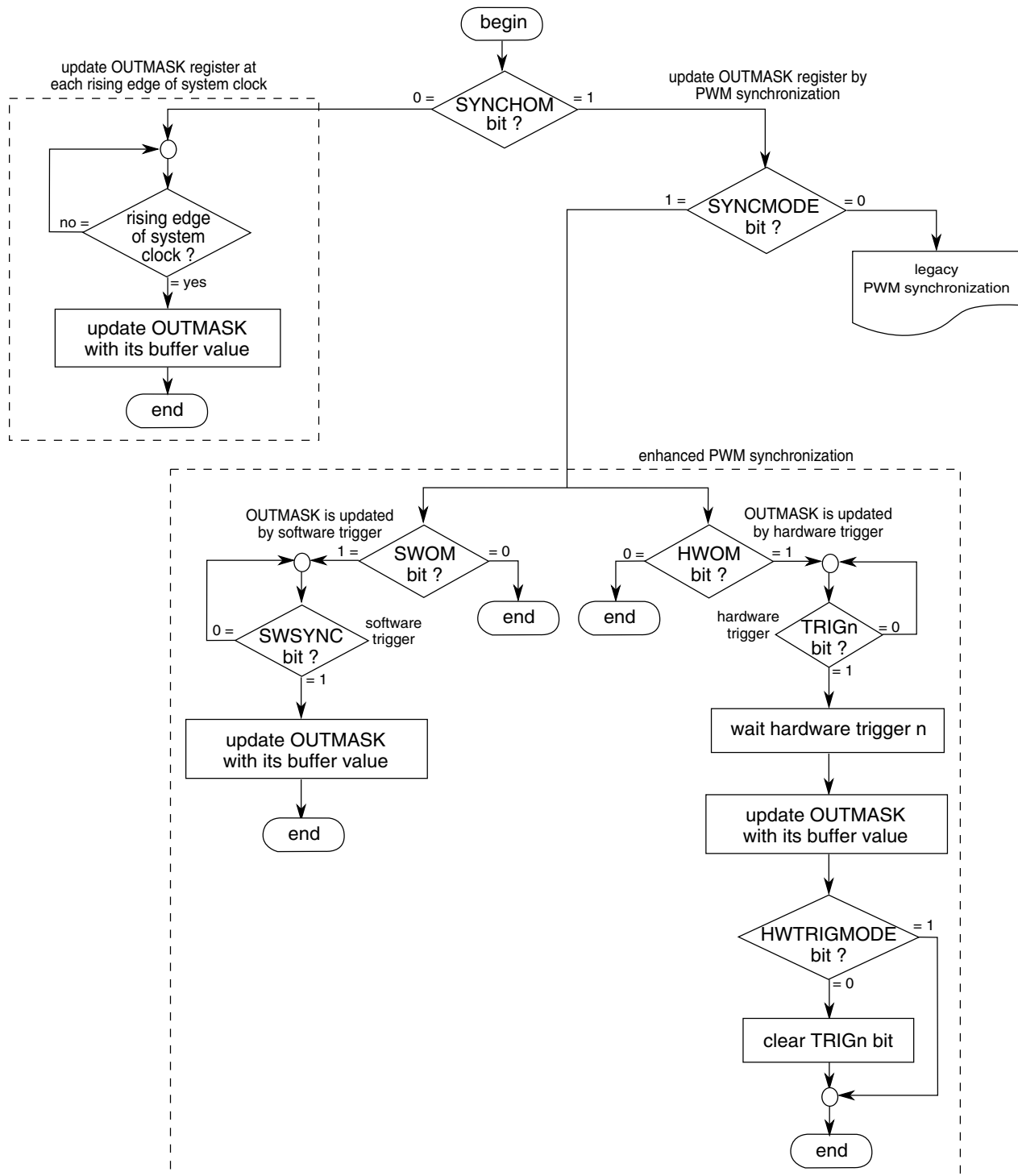
### 35.4.11.7 OUTMASK register synchronization

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:



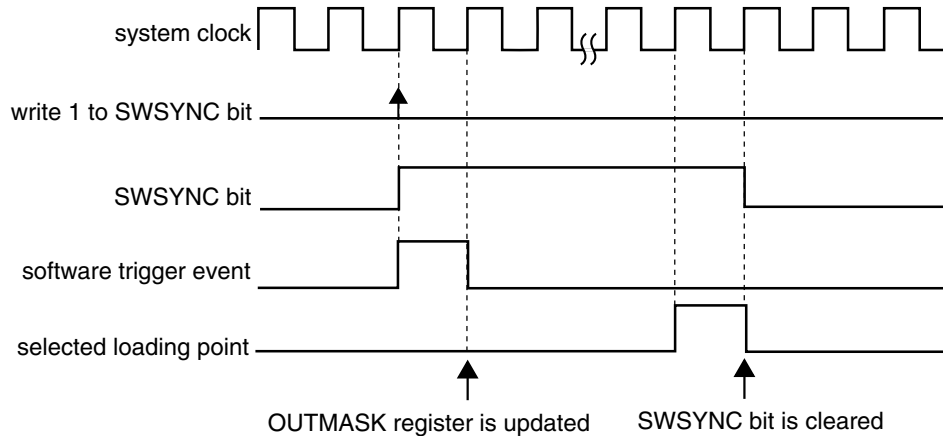


**Figure 35-215. OUTMASK register synchronization flowchart**

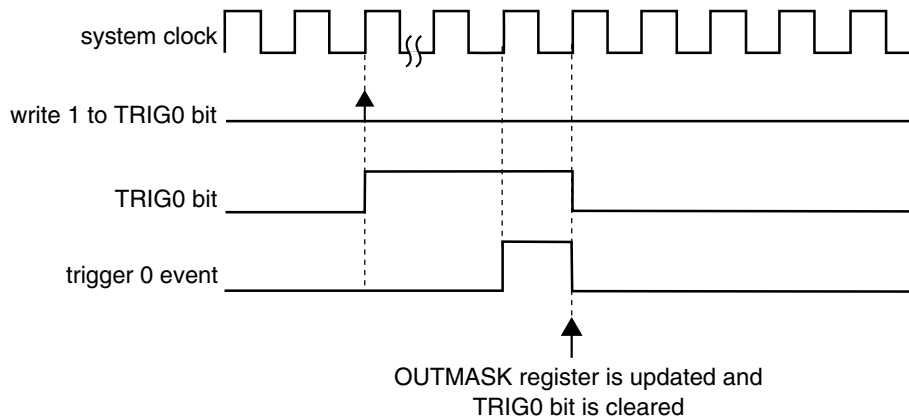
In the case of legacy PWM synchronization, the OUTMASK register synchronization depends on PWMSYNC bit according to the following description.

## Functional 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.

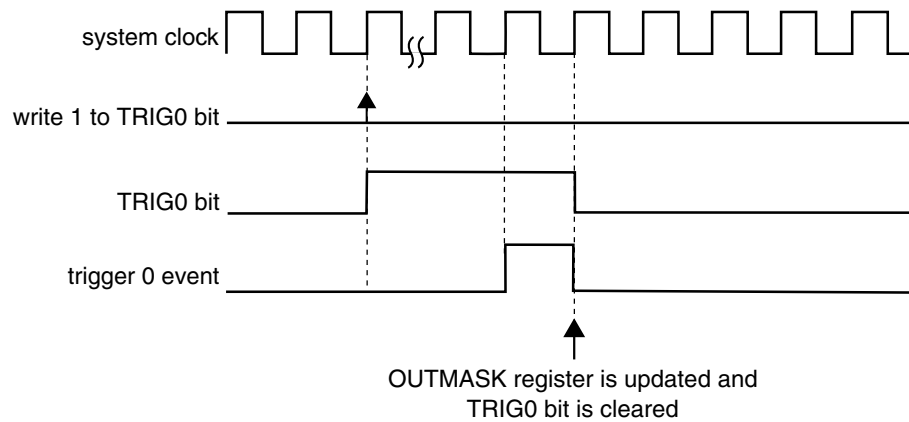


**Figure 35-216. OUTMASK synchronization with (SYNCMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0) and software trigger was used**



**Figure 35-217. 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 35-218. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 1), and a hardware trigger was used**

### 35.4.11.8 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.

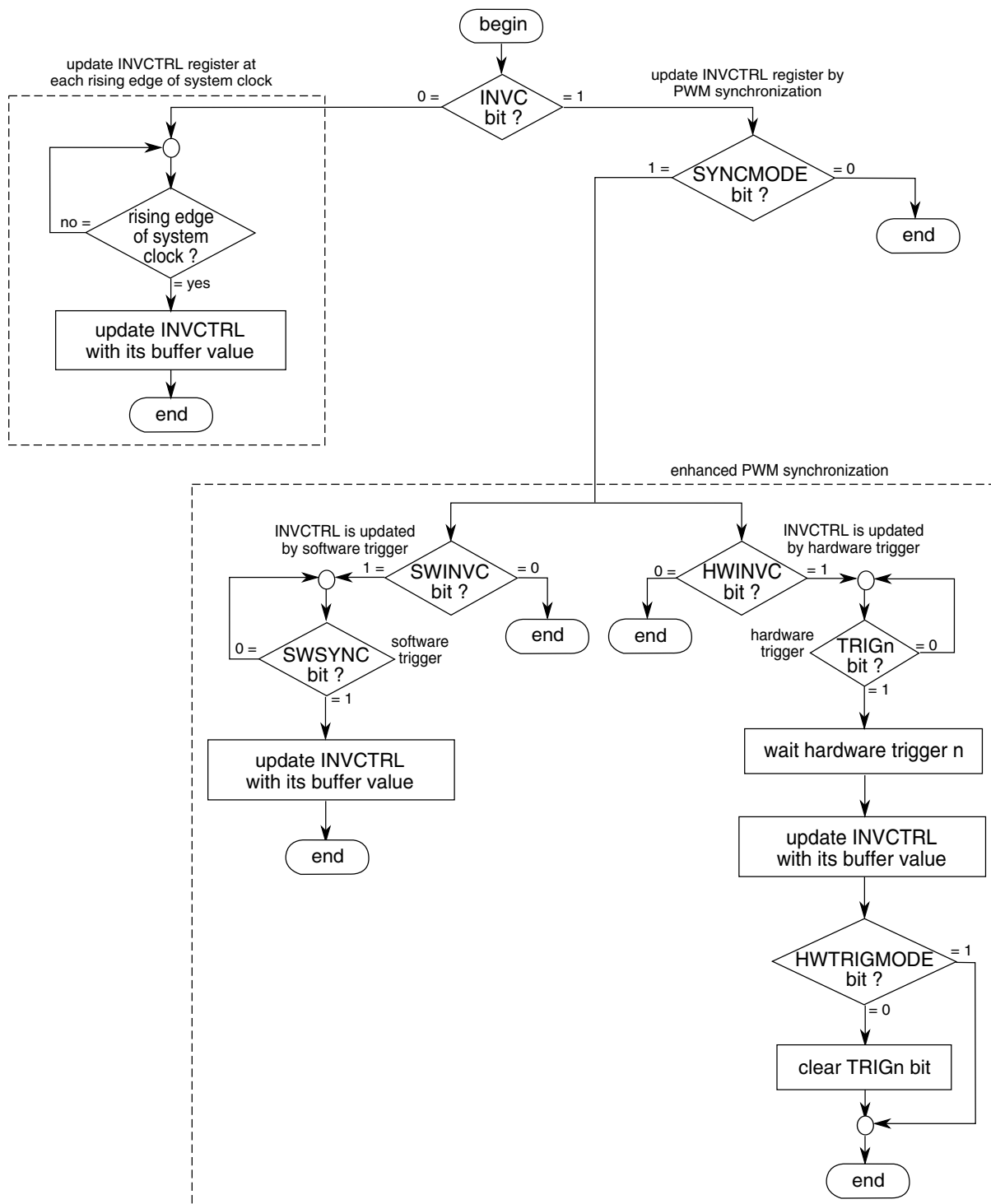


Figure 35-219. INVCTRL register synchronization flowchart

### 35.4.11.9 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.

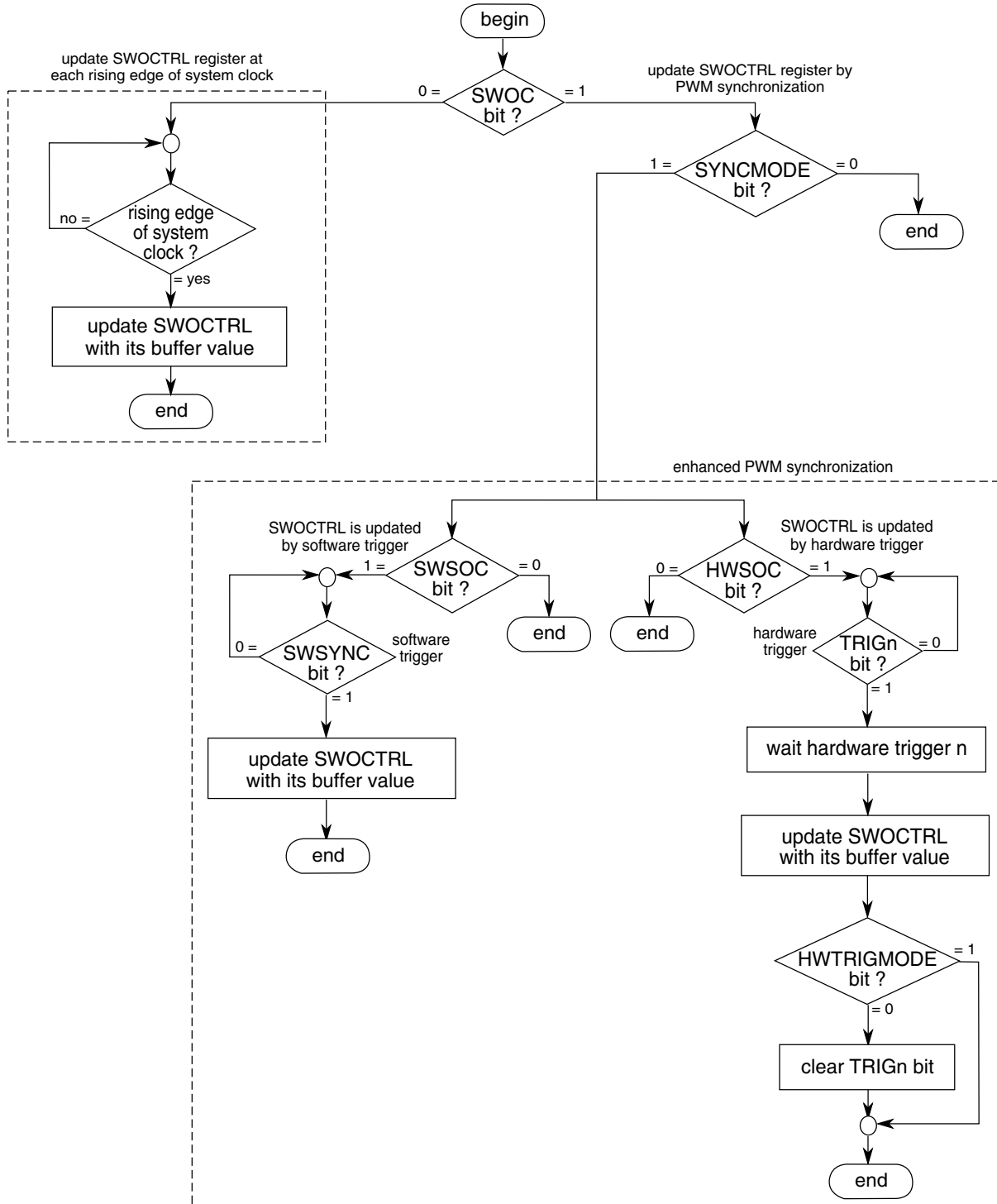
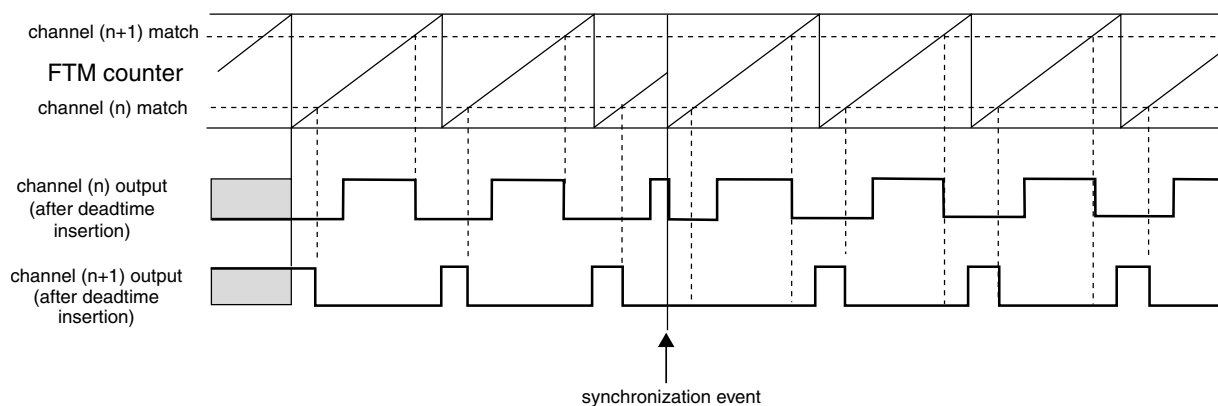


Figure 35-220. SWOCTRL register synchronization flowchart

### 35.4.11.10 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.

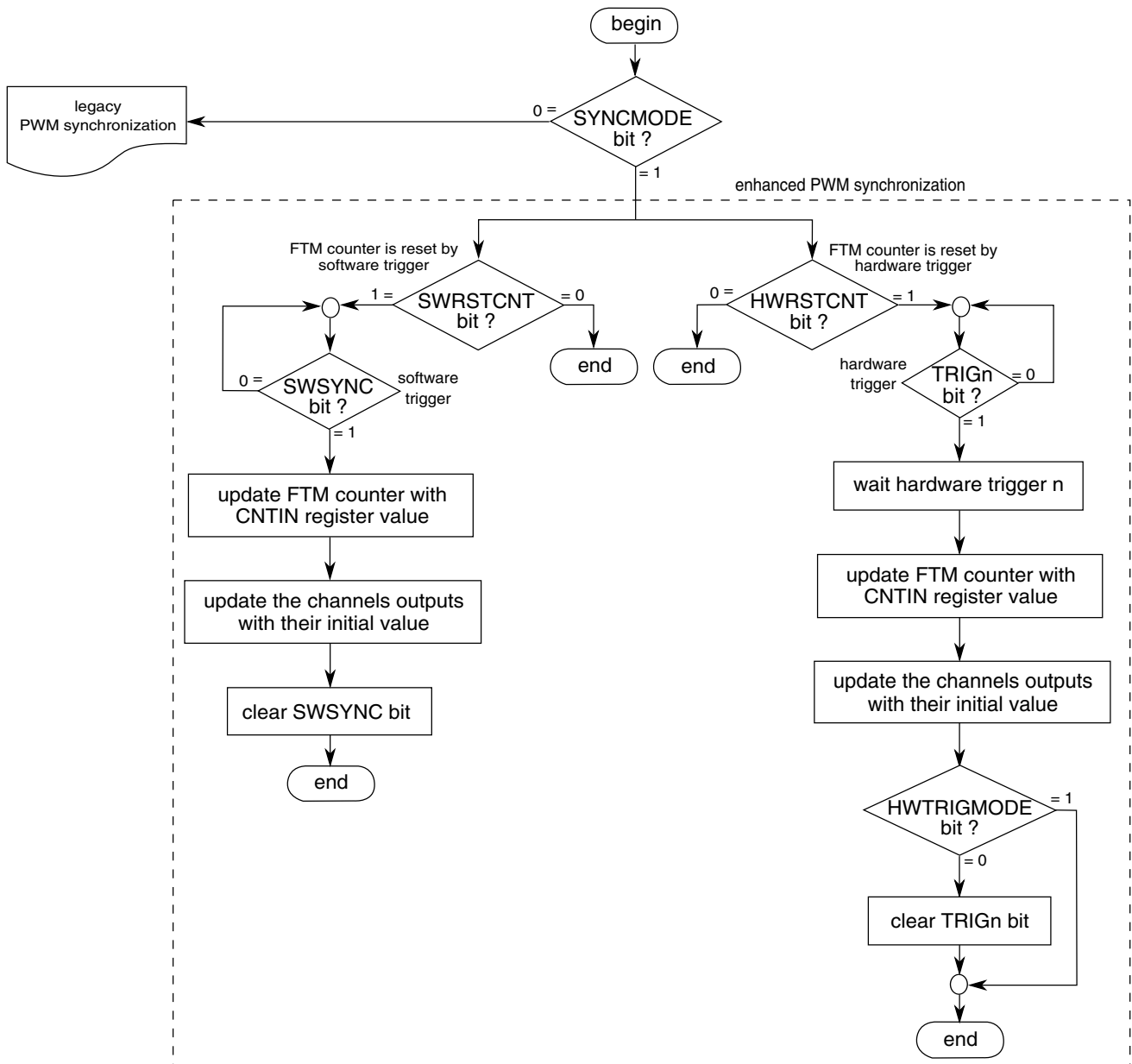
The following figure shows the FTM counter synchronization. Note that after the synchronization event occurs, the channel (n) is set to its initial value and the channel (n + 1) is not set to its initial value due to a specific timing of this figure in which the deadtime insertion prevents this channel 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 35-221. 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.

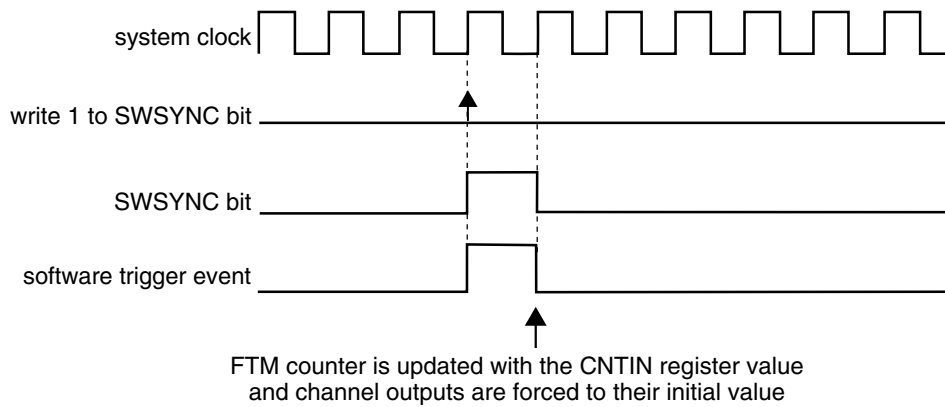


**Figure 35-222. FTM counter synchronization flowchart**

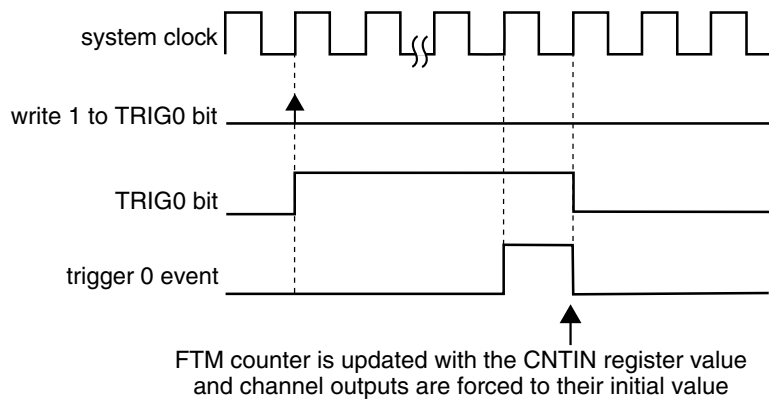
In the case of legacy PWM synchronization, the FTM counter synchronization depends on REINIT and PWMSYNC bits according to the following description.

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.

Functional description

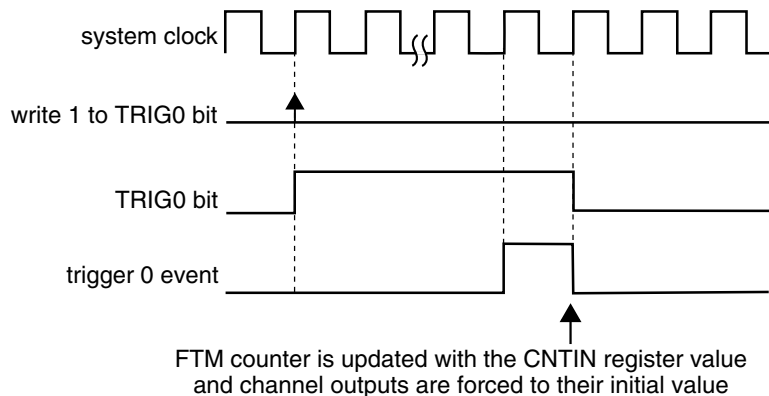


**Figure 35-223. FTM counter synchronization with (SYNCMODE = 0), (REINIT = 1), (PWMSYNC = 0), and software trigger was used**



**Figure 35-224. 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 35-225. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 1), and a hardware trigger was used**



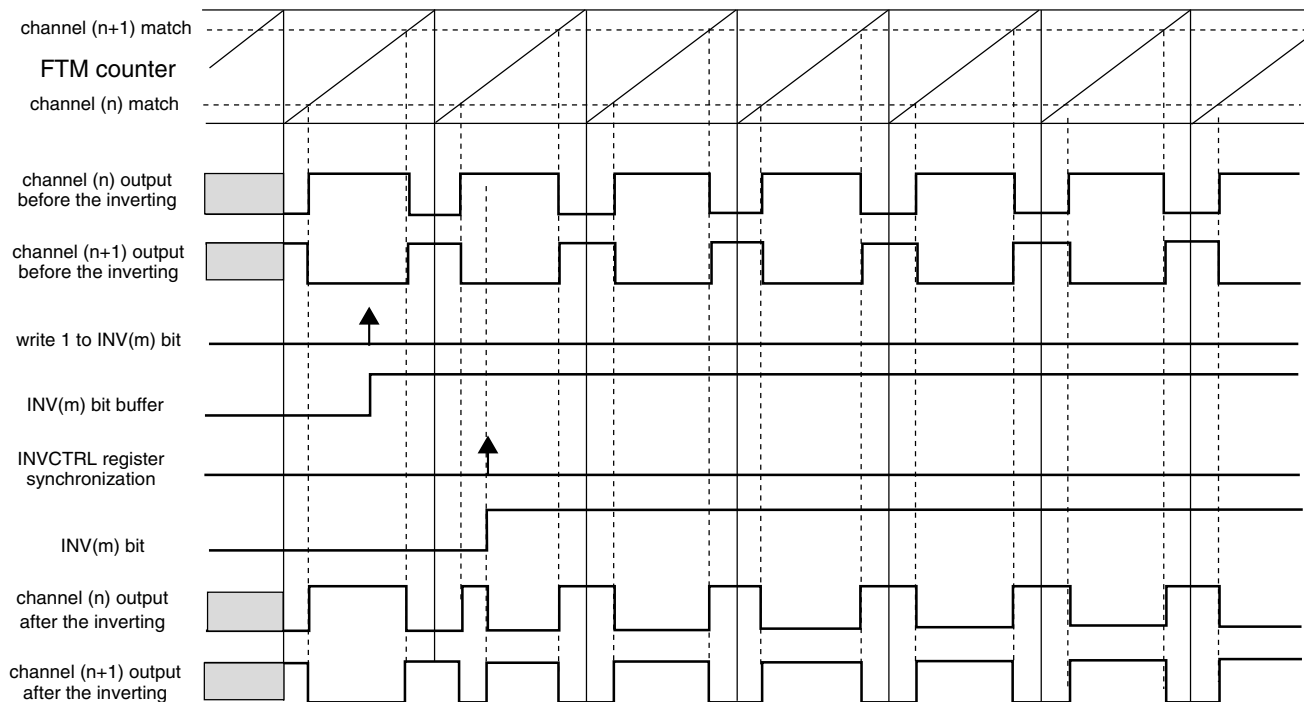
### 35.4.12 Inverting

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
- INV<sub>m</sub> = 1 (where m represents a channel pair)

The INV<sub>m</sub> 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.



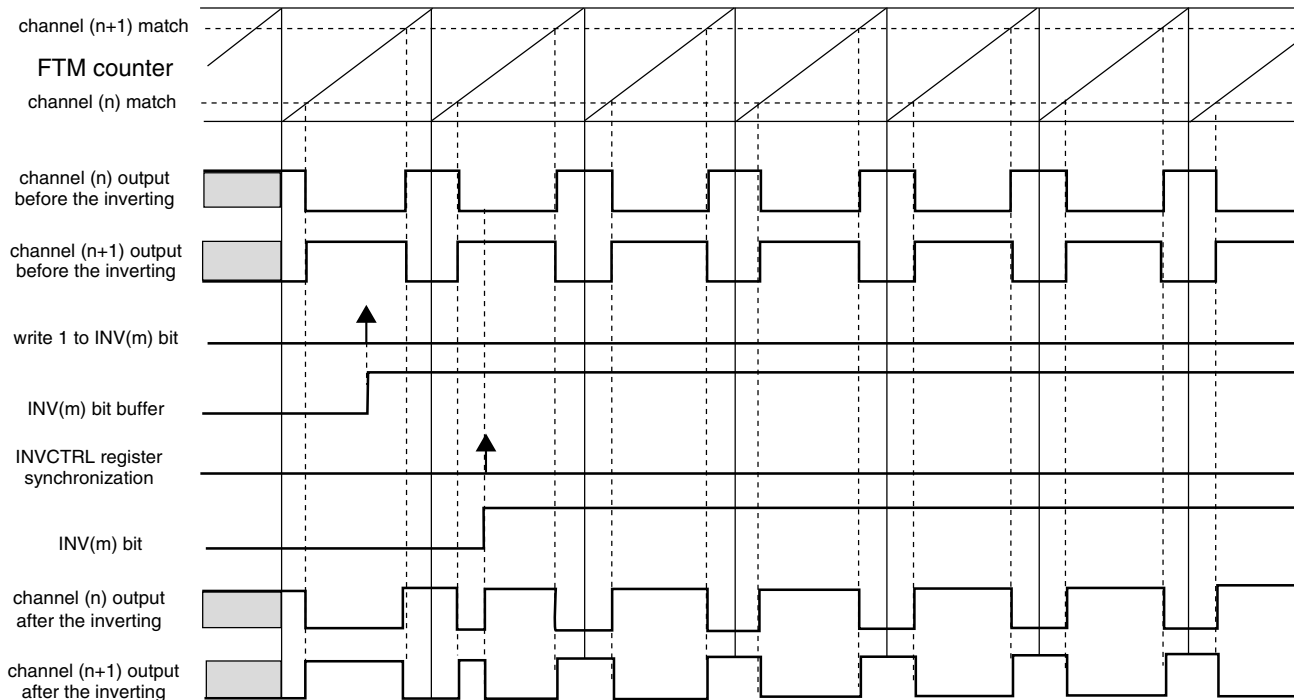
NOTE

INV<sub>m</sub> bit selects the inverting to the pair channels (n) and (n+1).

**Figure 35-226. Channels (n) and (n+1) outputs after the inverting in High-True (ELSnB:ELSnA = 1:0) Combine mode**

## Functional description

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 35-227. 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.

## 35.4.13 Software output control

The software output control forces the channel output according to software defined values at a specific time in the PWM generation.

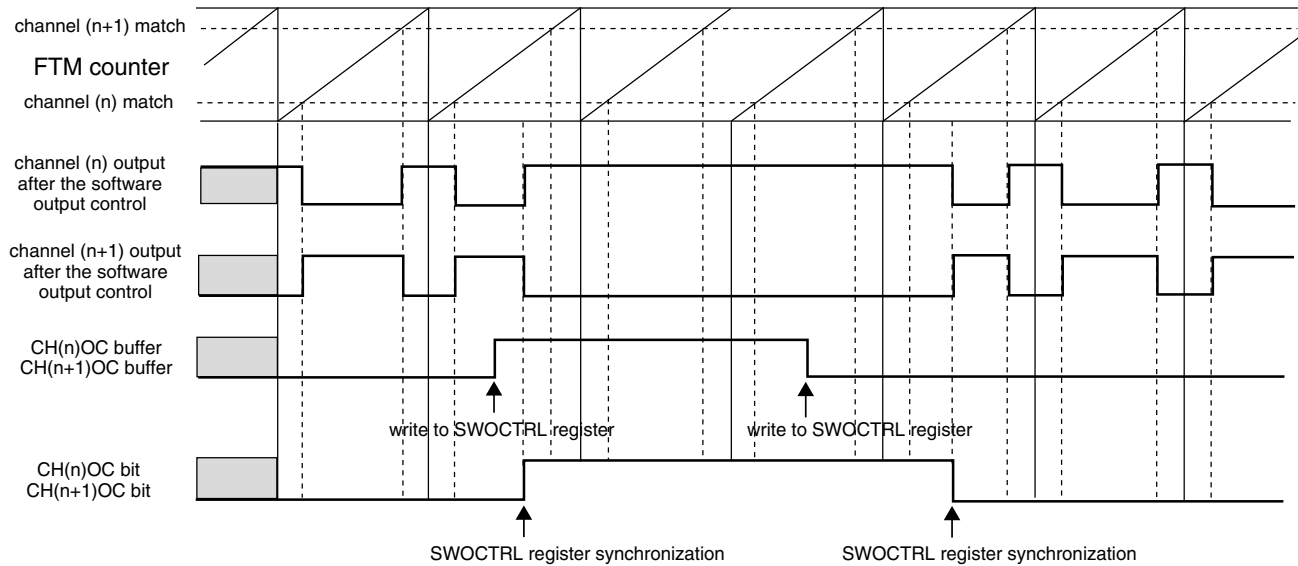
The software output control is selected when:

- 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 35-228. 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 35-246. Software output 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 35-247. Software output 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).

**35.4.14 Deadtime insertion**

The deadtime insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non-zero).

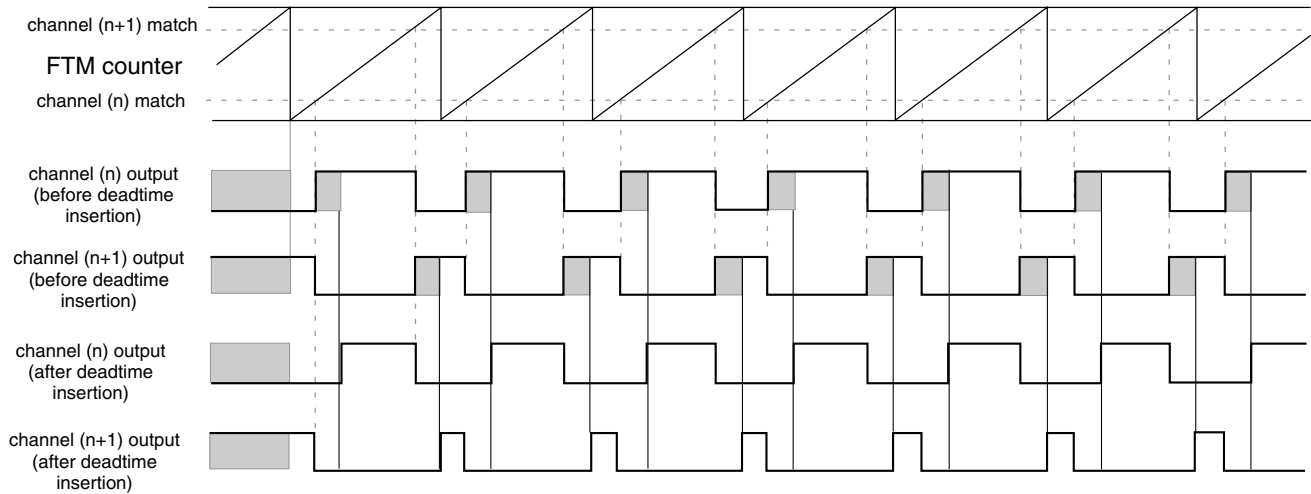
DEADTIME register defines the deadtime delay that can be used for all FTM channels. The DTPS[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the deadtime modulo, that is, the number of the deadtime prescaler clocks.

The deadtime delay insertion ensures that no two complementary signals (channels (n) and (n+1)) drive the active state at the same time.

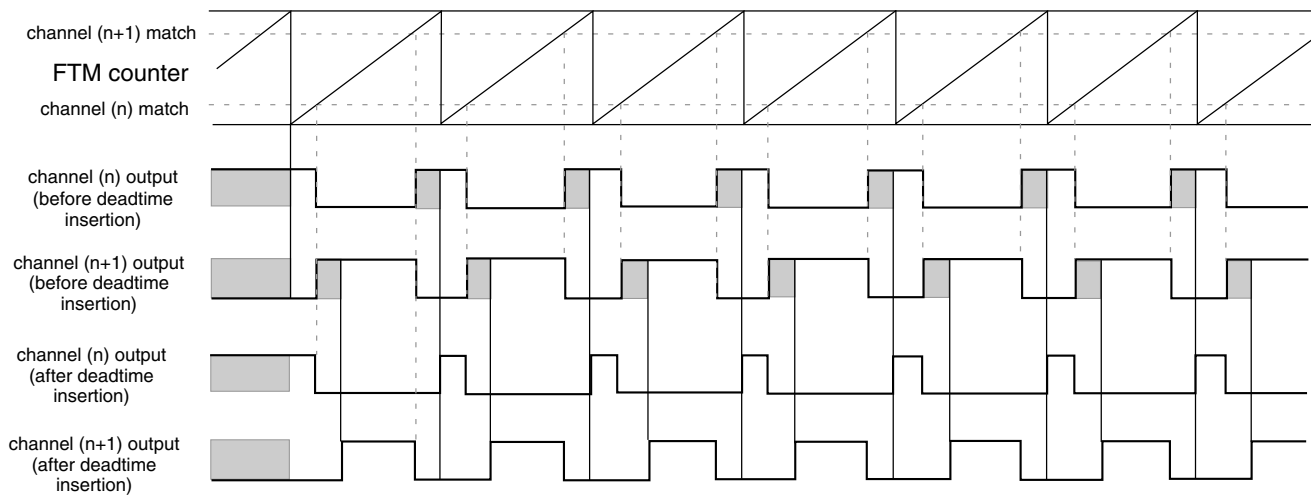
If POL(n) = 0, POL(n+1) = 0, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the low value until the end of the deadtime delay when the channel (n) output is set. Similarly, when the channel (n+1) match (FTM counter = C(n+1)V) 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) output is cleared.



**Figure 35-229. Deadtime insertion with  $ELSnB:ELSnA = 1:0$ ,  $POL(n) = 0$ , and  $POL(n+1) = 0$**



**Figure 35-230. Deadtime insertion with  $ELSnB:ELSnA = X:1$ ,  $POL(n) = 0$ , and  $POL(n+1) = 0$**

### NOTE

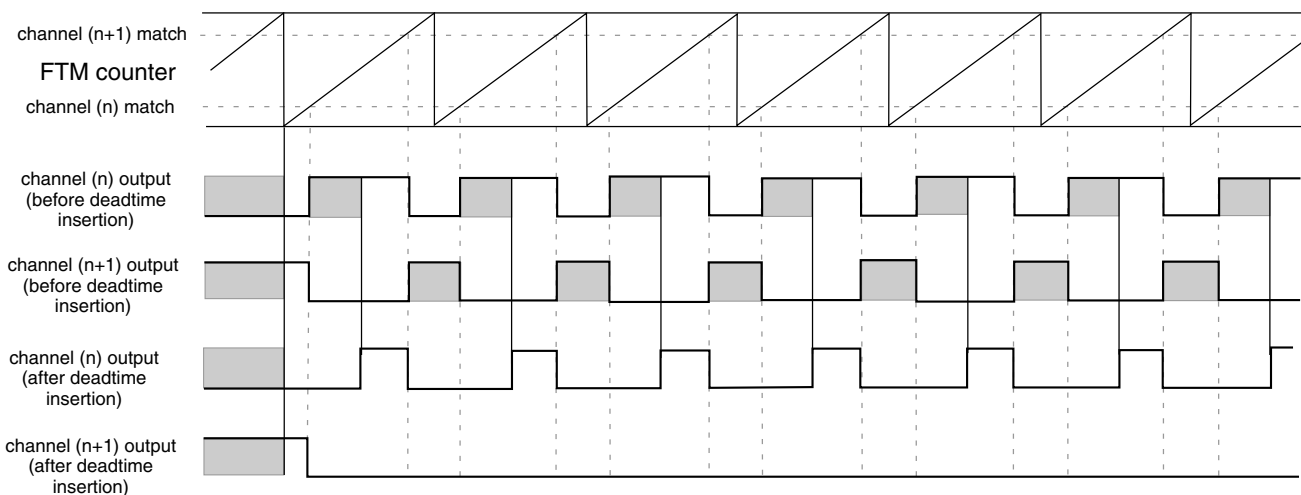
- The deadtime feature must be used only in Complementary mode.
- The deadtime feature is not available in Output Compare mode.

### 35.4.14.1 Deadtime insertion corner cases

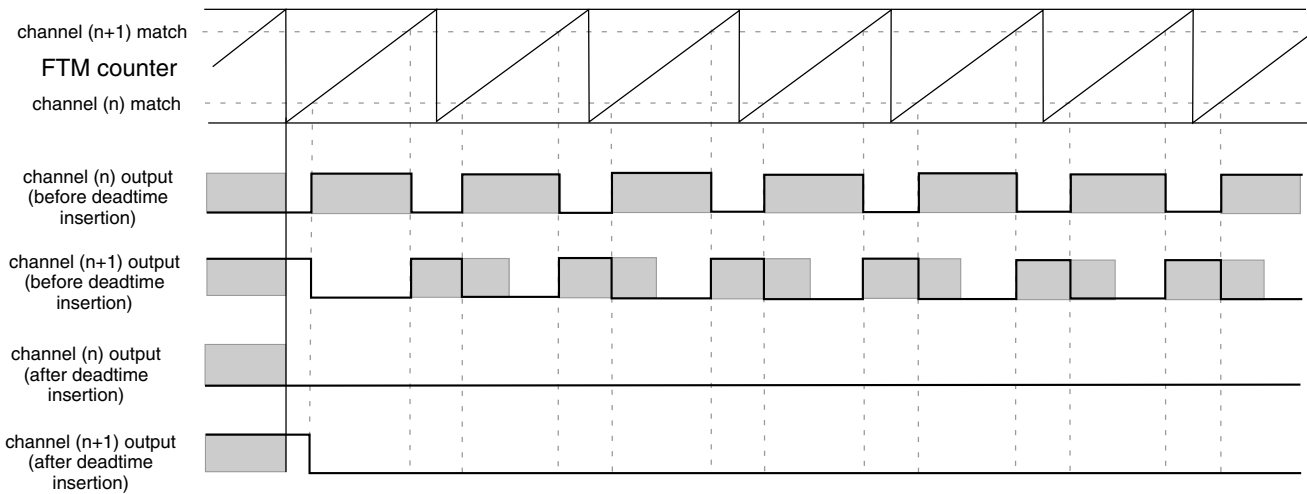
If (PS[2:0] is cleared), (DTPS[1:0] = 0:0 or DTPS[1:0] = 0:1):

- and the deadtime delay is greater than or equal to the channel (n) duty cycle ( $(C(n+1)V - C(n)V) \times \text{system clock}$ ), 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 ( $(\text{MOD} - \text{CNTIN} + 1 - (C(n+1)V - C(n)V)) \times \text{system clock}$ ), 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 35-231. 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**



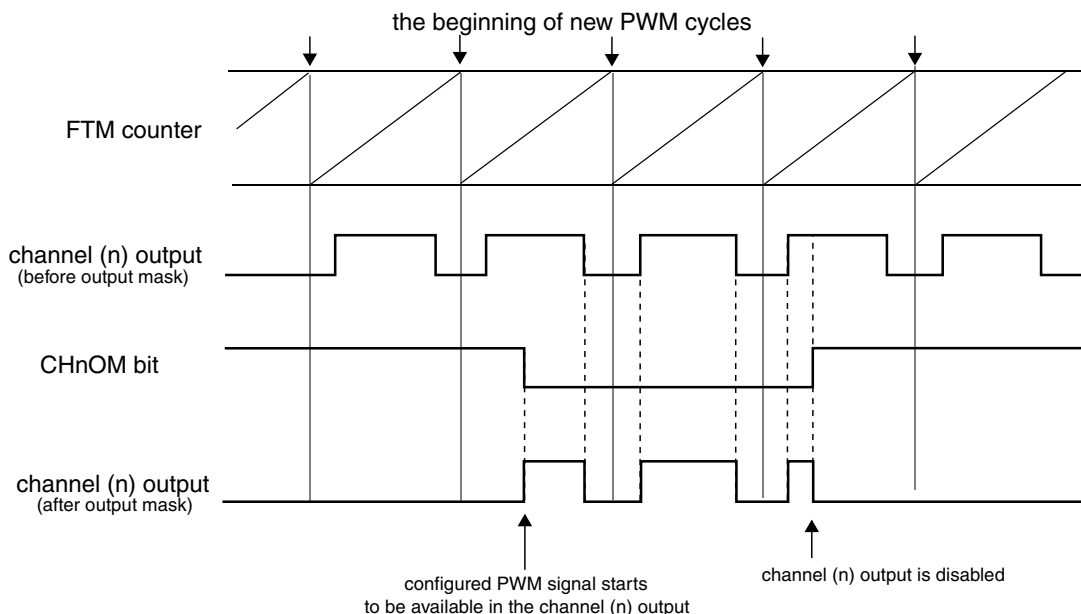
**Figure 35-232. Example of the deadtime insertion ( $ELSnB:ELSnA = 1:0$ ,  $POL(n) = 0$ , and  $POL(n+1) = 0$ ) when the deadtime delay is comparable to channels (n) and (n+1) duty cycle**

### 35.4.15 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 35-233. Output mask with POLn = 0**

The following table shows the output mask result before the polarity control.

**Table 35-248. 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	

### 35.4.16 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

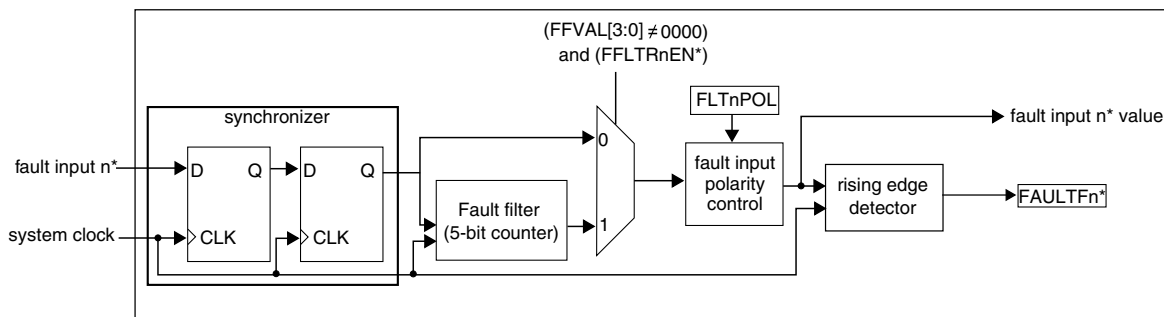


counter continues to increment. If the 5-bit counter overflows, that is, the counter exceeds the value of the FFVAL[3:0] bits, the new fault input n value is validated. It is then transmitted as a pulse edge to the edge detector.

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 ( $\times$  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]  $\neq$  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 35-234. 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.

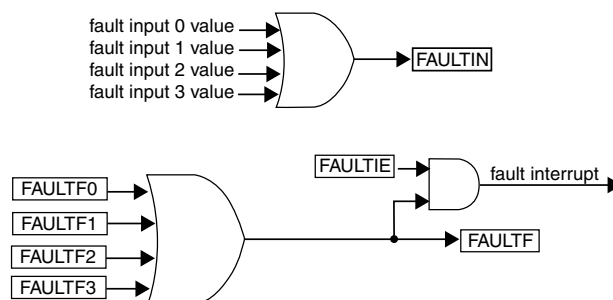


Figure 35-235. FAULTF and FAULTIN bits and fault interrupt

## Functional description

If the fault control is enabled ( $\text{FAULTM}[1:0] \neq 0:0$ ), a fault condition has occurred and ( $\text{FAULTEN} = 1$ ), then outputs are forced to their safe values:

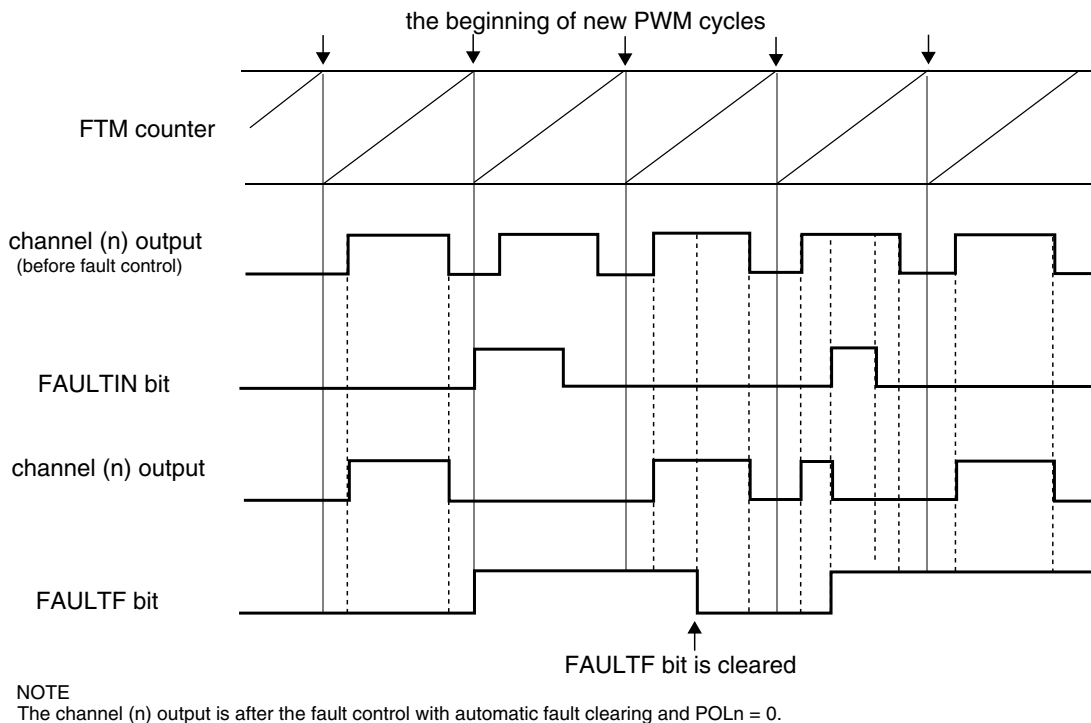
- Channel (n) output takes the value of  $\text{POL}(n)$
- Channel (n+1) takes the value of  $\text{POL}(n+1)$

The fault interrupt is generated when ( $\text{FAULTF} = 1$ ) and ( $\text{FAULTIE} = 1$ ). This interrupt request remains set until:

- Software clears the  $\text{FAULTF}$  bit by reading  $\text{FAULTF}$  bit as 1 and writing 0 to it
- Software clears the  $\text{FAULTIE}$  bit
- A reset occurs

### 35.4.16.1 Automatic fault clearing

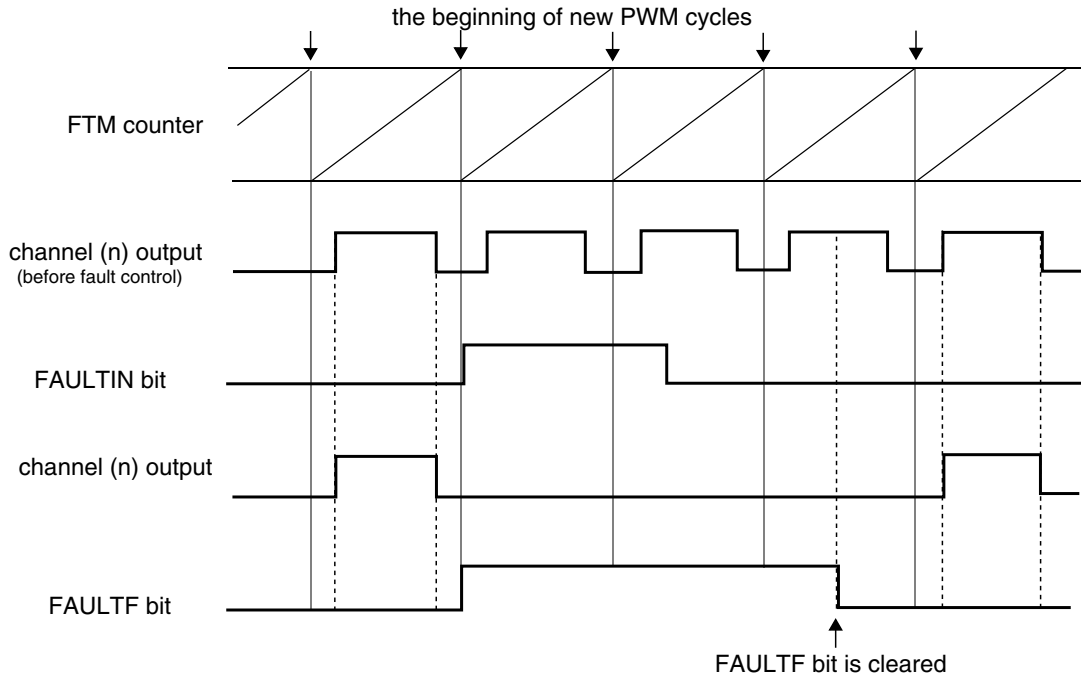
If the automatic fault clearing is selected ( $\text{FAULTM}[1:0] = 1:1$ ), then the channels output disabled by fault control is again enabled when the fault input signal ( $\text{FAULTIN}$ ) returns to zero and a new PWM cycle begins. See the following figure.



**Figure 35-236. Fault control with automatic fault clearing**

### 35.4.16.2 Manual fault clearing

If the manual fault clearing is selected ( $\text{FAULTM}[1:0] = 0:1$  or  $1:0$ ), then the channels output disabled by fault control is again enabled when the  $\text{FAULTF}$  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  $\text{POLn} = 0$ .

**Figure 35-237. Fault control with manual fault clearing**

### 35.4.16.3 Fault inputs polarity control

The  $\text{FLTjPOL}$  bit selects the fault input  $j$  polarity, where  $j = 0, 1, 2, 3$ :

- If  $\text{FLTjPOL} = 0$ , the fault  $j$  input polarity is high, so the logical one at the fault input  $j$  indicates a fault.
- If  $\text{FLTjPOL} = 1$ , the fault  $j$  input polarity is low, so the logical zero at the fault input  $j$  indicates a fault.

## 35.4.17 Polarity control

The  $\text{POLn}$  bit selects the channel (n) output polarity:

## Functional description

- If  $POL_n = 0$ , the channel (n) output polarity is high, so the logical one is the active state and the logical zero is the inactive state.
- If  $POL_n = 1$ , the channel (n) output polarity is low, so the logical zero is the active state and the logical one is the inactive state.

### 35.4.18 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 35-249. 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 35-250. 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

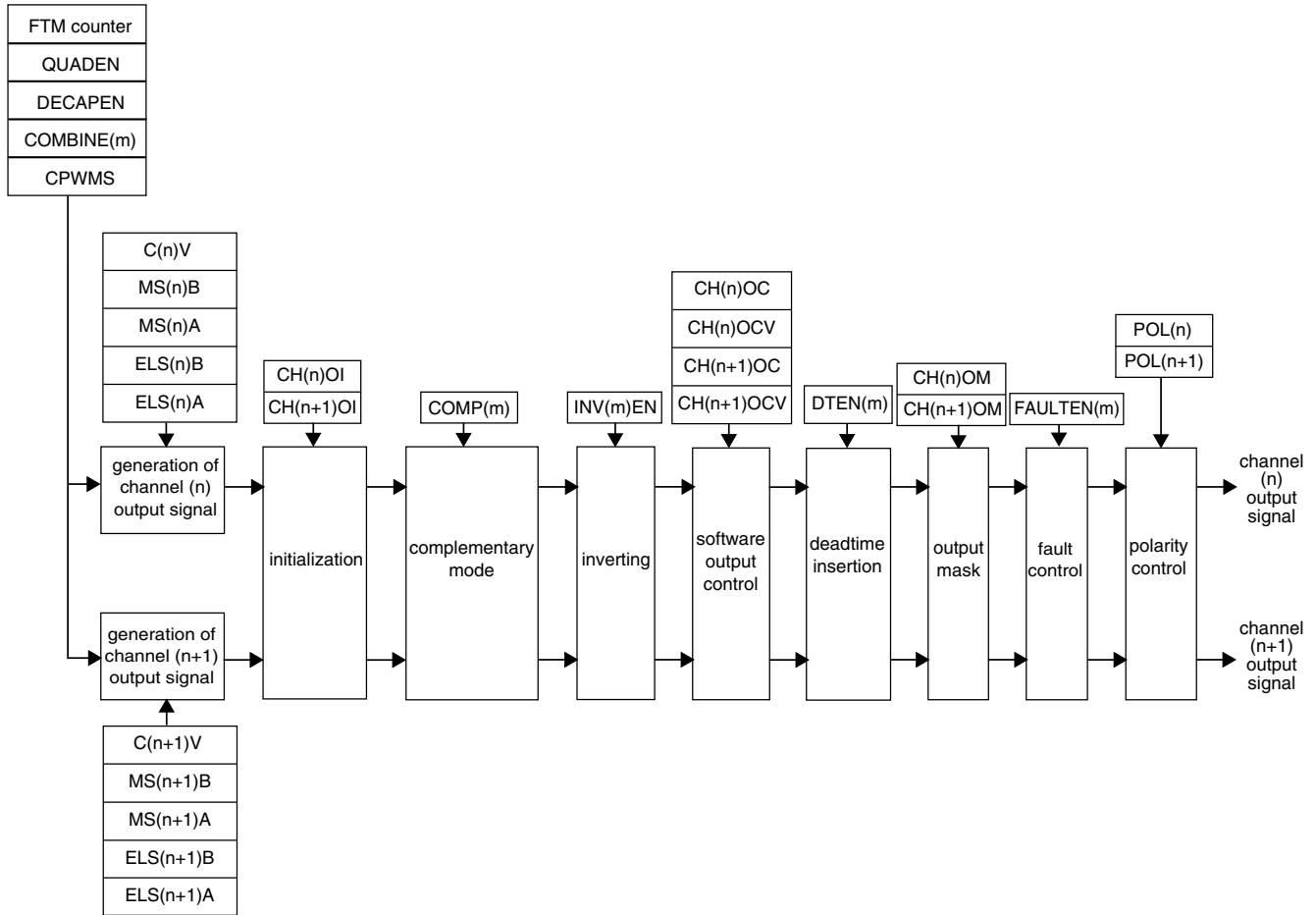
#### Note

The initialization feature must be used only with disabled FTM counter. See the description of the CLKS field in the Status and Control register.

### 35.4.19 Features priority

The following figure shows the priority of the features used at the generation of channels (n) and (n+1) outputs signals.

pair channels (m) - channels (n) and (n+1)

**NOTE**

The channels (n) and (n+1) are in output compare, EPWM, CPWM or combine modes.

**Figure 35-238. 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.

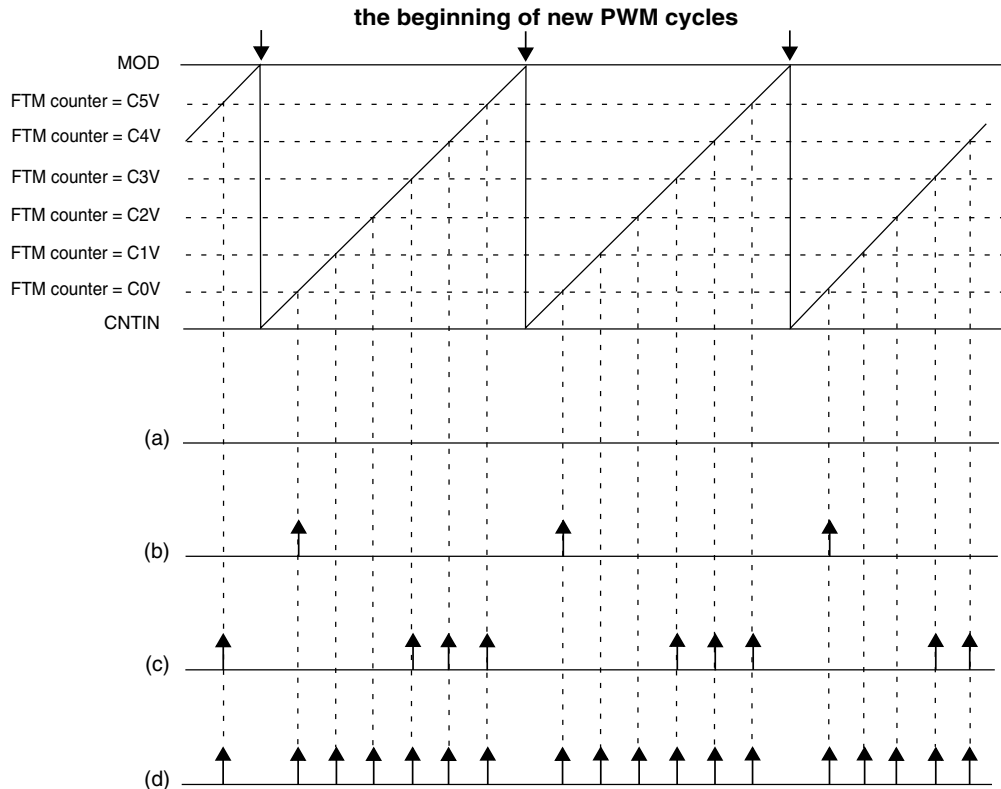
### 35.4.20 Channel trigger output

If  $CH_jTRIG = 1$ , where  $j = 0, 1, 2, 3, 4, \text{ or } 5$ , 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 that is used for on-chip modules.

## Functional description

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 35-239. Channel match trigger**

## 35.4.21 Initialization trigger

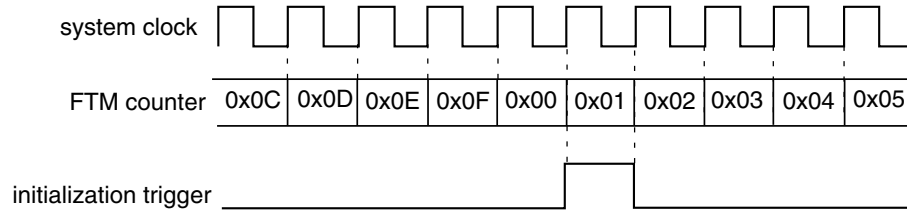
If INITTRIGEN = 1, then the FTM generates a trigger when the FTM counter is updated with the CNTIN register value in the following cases.

- 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.

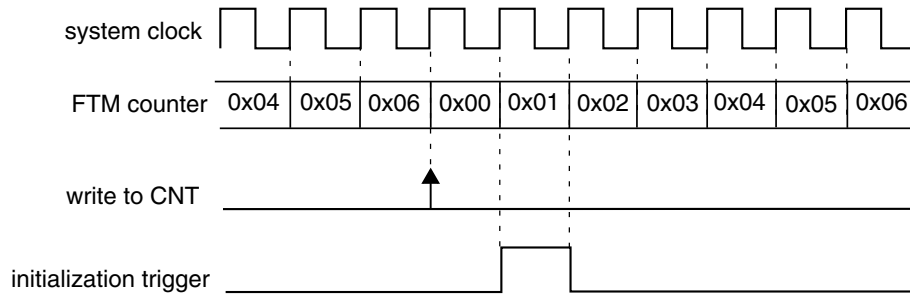
The following figures show these cases.

CNTIN = 0x0000  
 MOD = 0x000F  
 CPWMS = 0



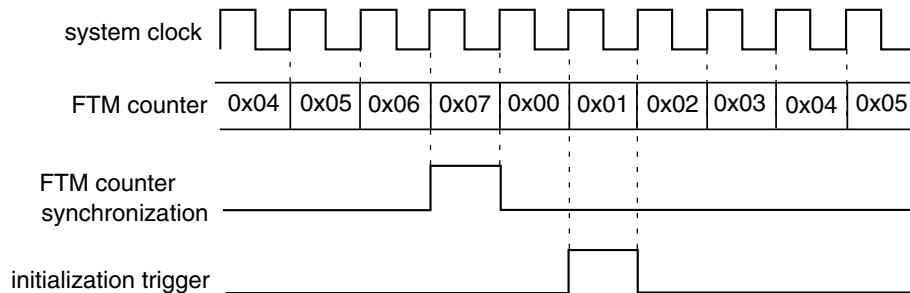
**Figure 35-240. Initialization trigger is generated when the FTM counting achieves the CNTIN register value**

CNTIN = 0x0000  
 MOD = 0x000F  
 CPWMS = 0



**Figure 35-241. Initialization trigger is generated when there is a write to CNT register**

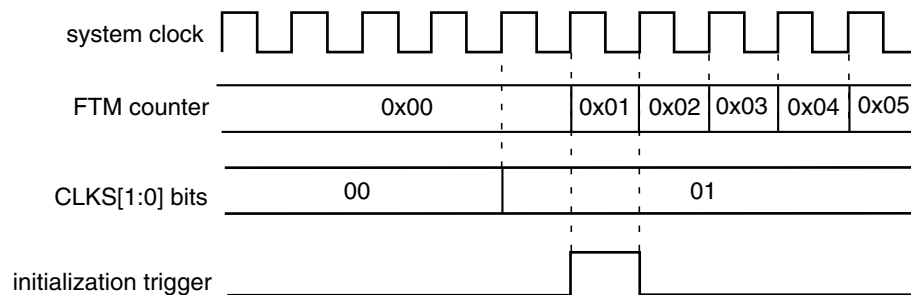
CNTIN = 0x0000  
 MOD = 0x000F  
 CPWMS = 0



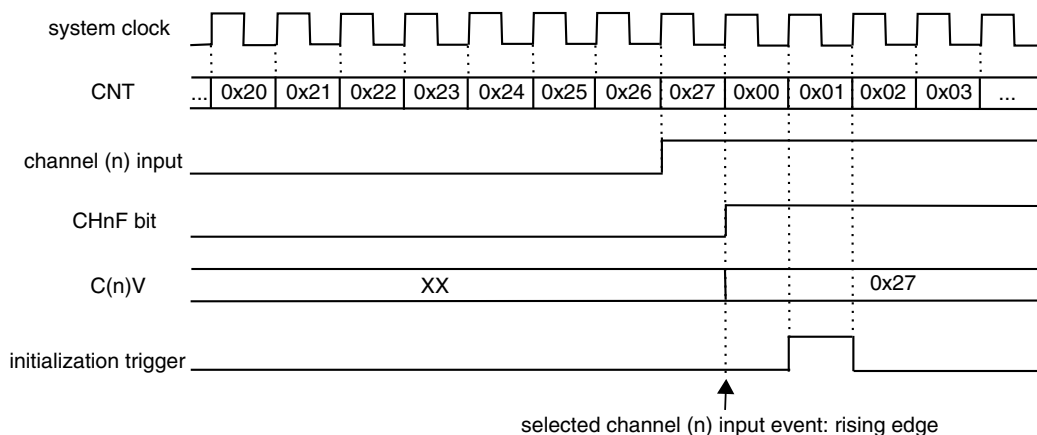
**Figure 35-242. Initialization trigger is generated when there is the FTM counter synchronization**

## Functional description

CNTIN = 0x0000  
 MOD = 0x000F  
 CPWMS = 0



**Figure 35-243. 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 35-244. 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.

### 35.4.22 Capture Test mode

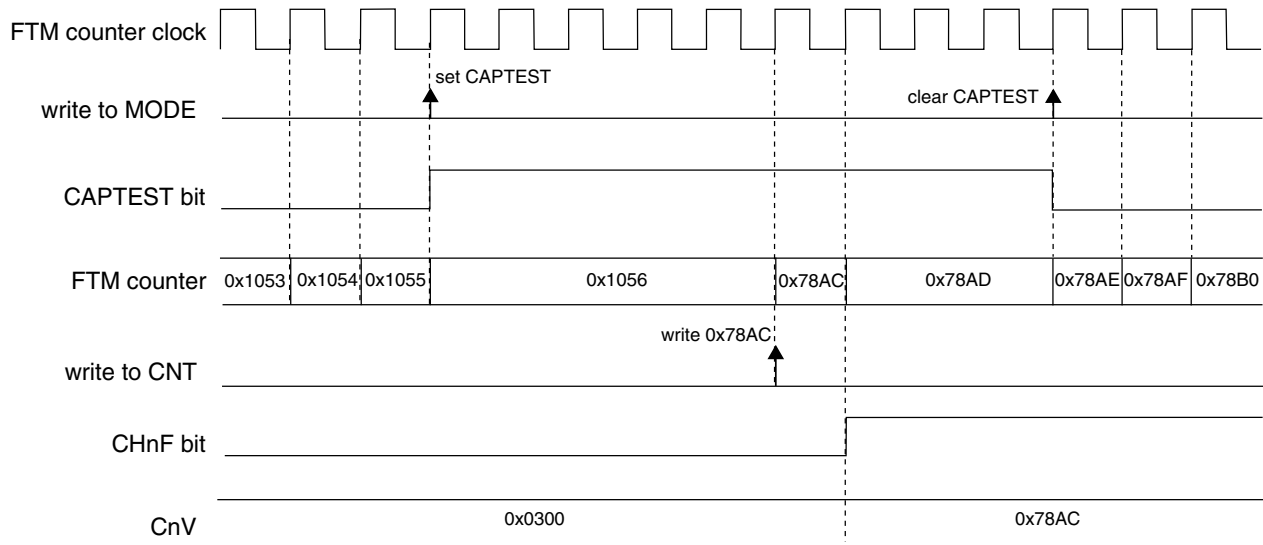
The Capture Test mode allows to test the CnV registers, the FTM counter and the interconnection logic between the FTM counter and CnV registers.

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 35-245. Capture Test mode**

### 35.4.23 DMA

The channel generates a DMA transfer request according to DMA and CHnIE bits. See the following table.

**Table 35-251. 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.

Table continues on the next page...

**Table 35-251. Channel DMA transfer request (continued)**

DMA	CHnIE	Channel DMA Transfer Request	Channel Interrupt
1	1	The channel DMA transfer request is generated if (CHnF = 1).	The channel interrupt is not generated.

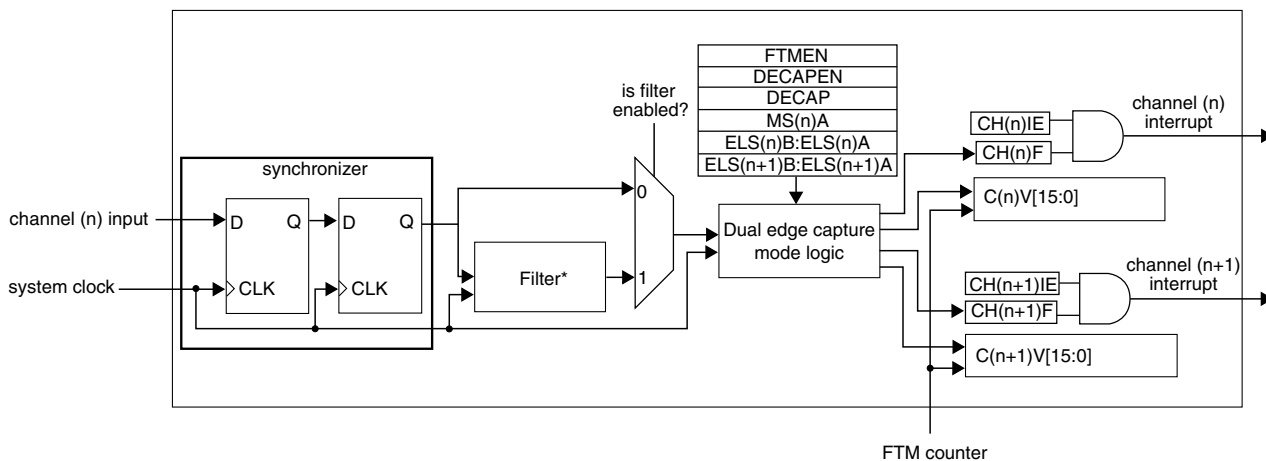
If DMA = 1, the CHnF bit is cleared either by channel DMA transfer done or reading CnSC while CHnF is set and then writing a zero to CHnF bit according to CHnIE bit. See the following table.

**Table 35-252. 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.

### 35.4.24 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.



\* Filtering function for dual edge capture mode is only available in the channels 0 and 2

**Figure 35-246. 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](#).

#### 35.4.24.1 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.

### 35.4.24.2 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.

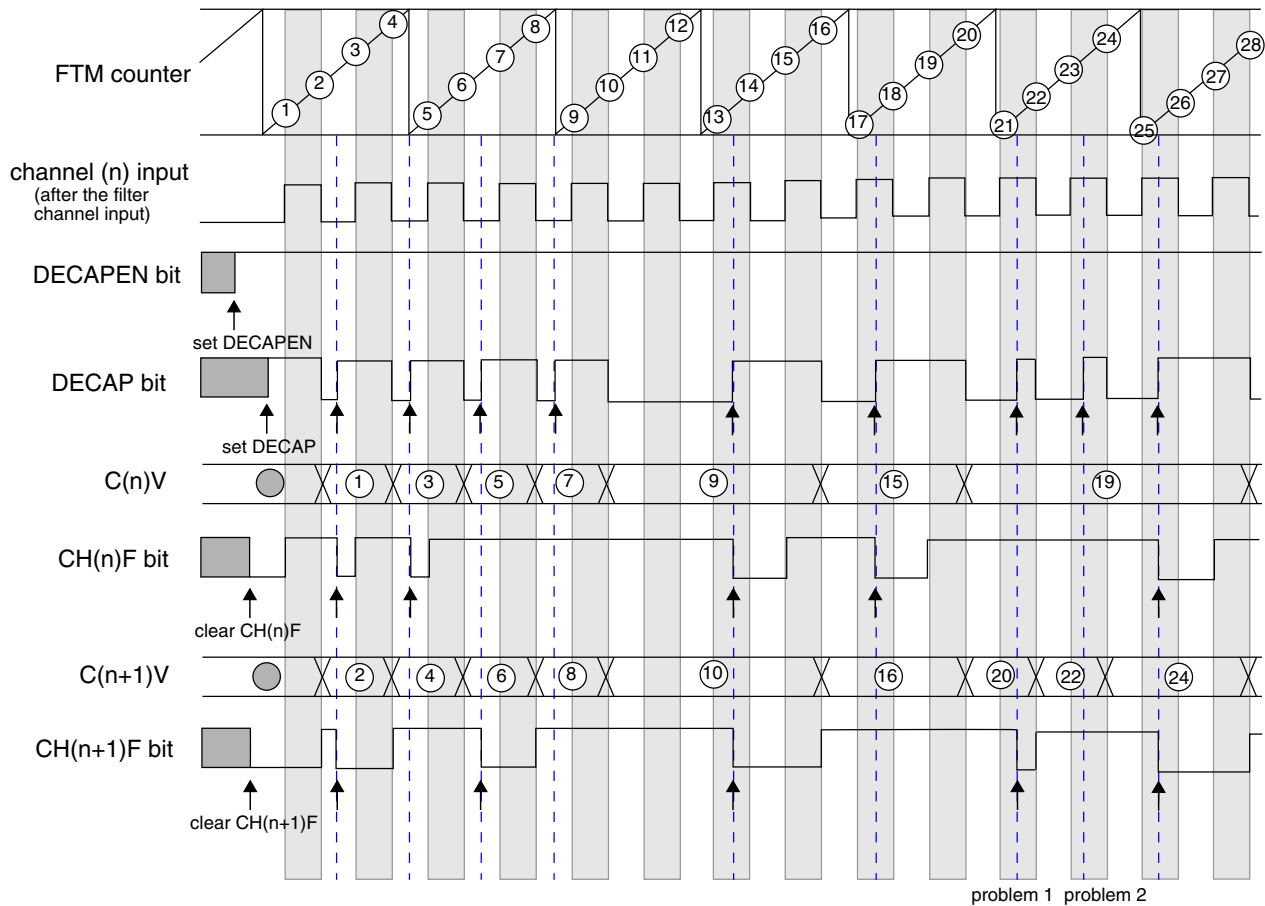
### 35.4.24.3 Pulse width measurement

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

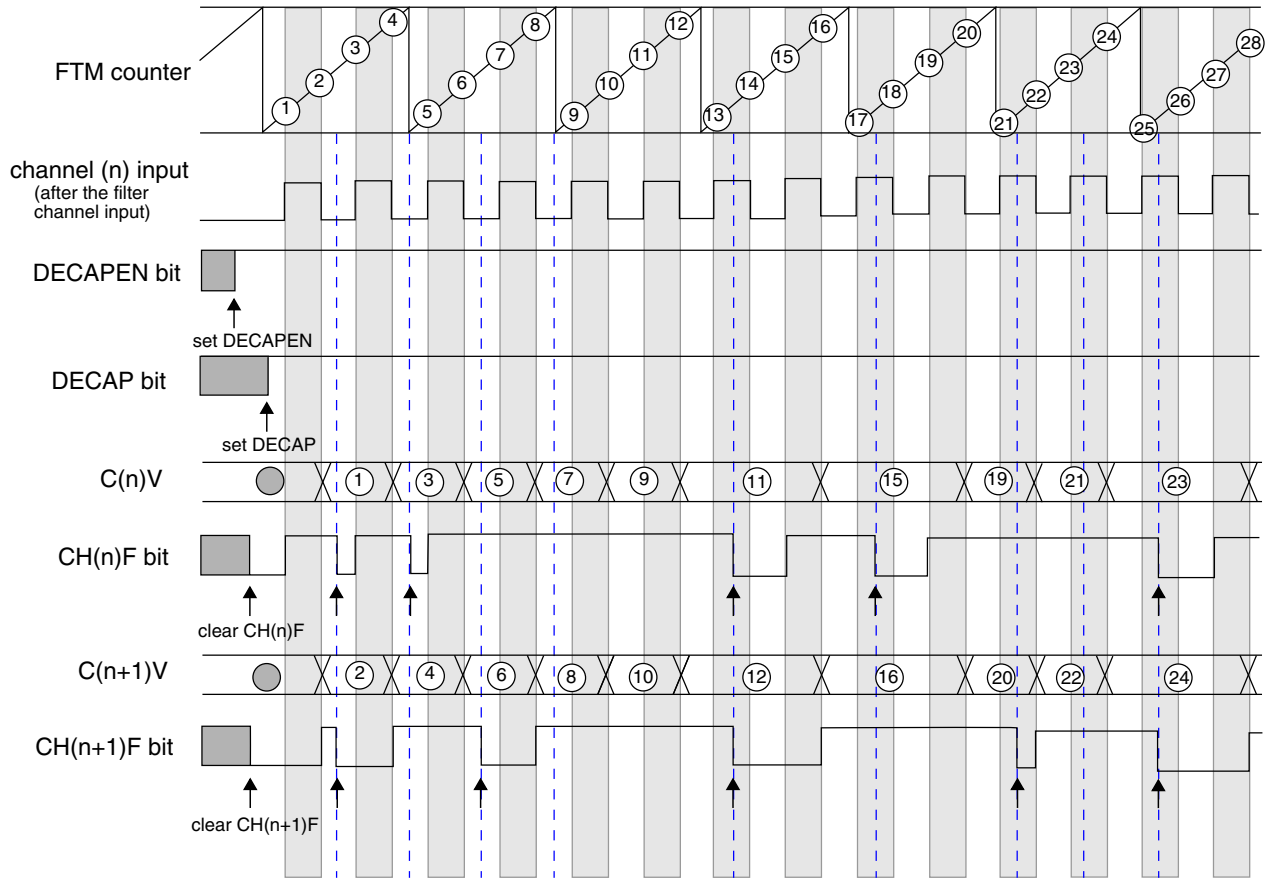
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

**Figure 35-247. 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

## Functional description

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.

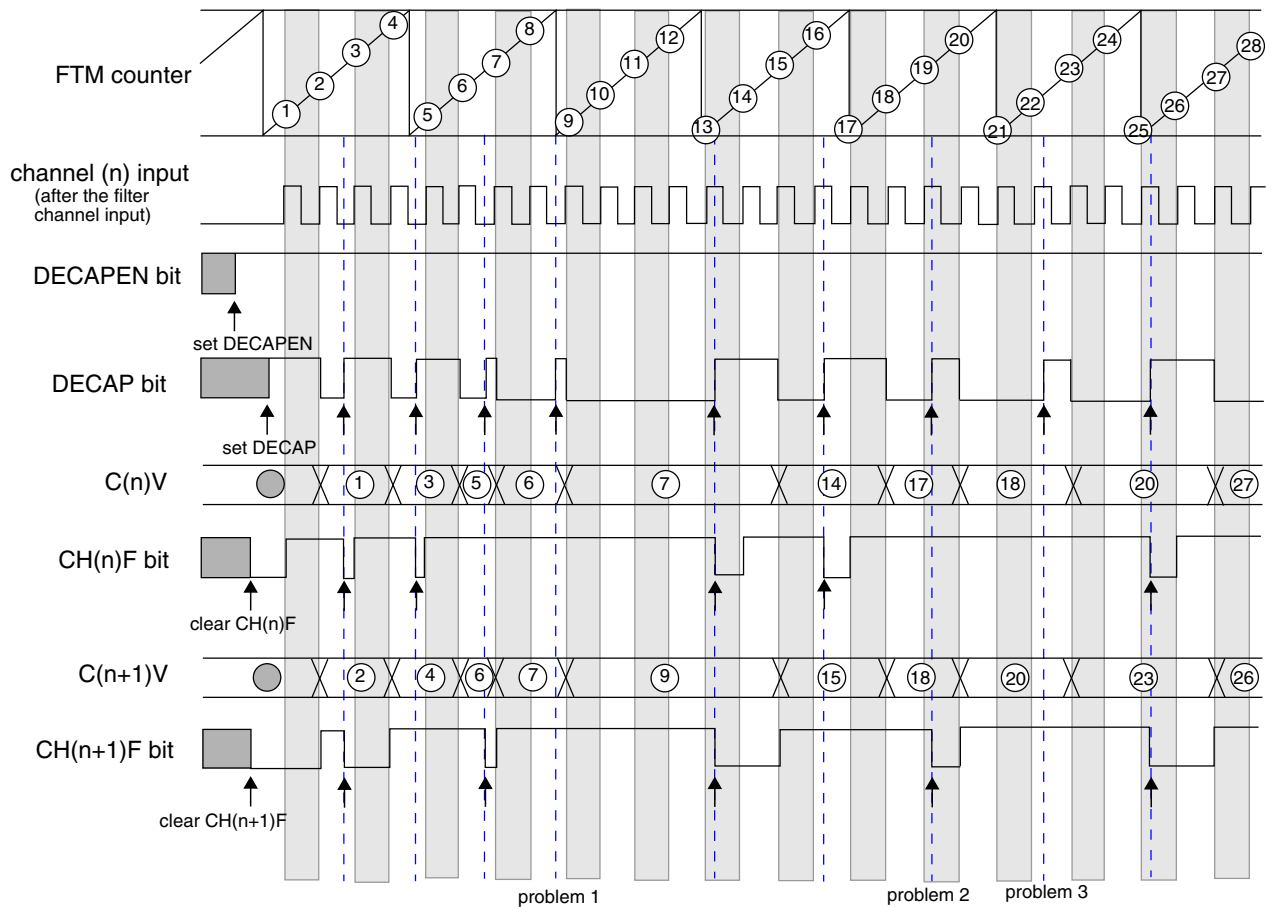
**Figure 35-248. Dual Edge Capture – Continuous mode for positive polarity pulse width measurement**

### 35.4.24.4 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.



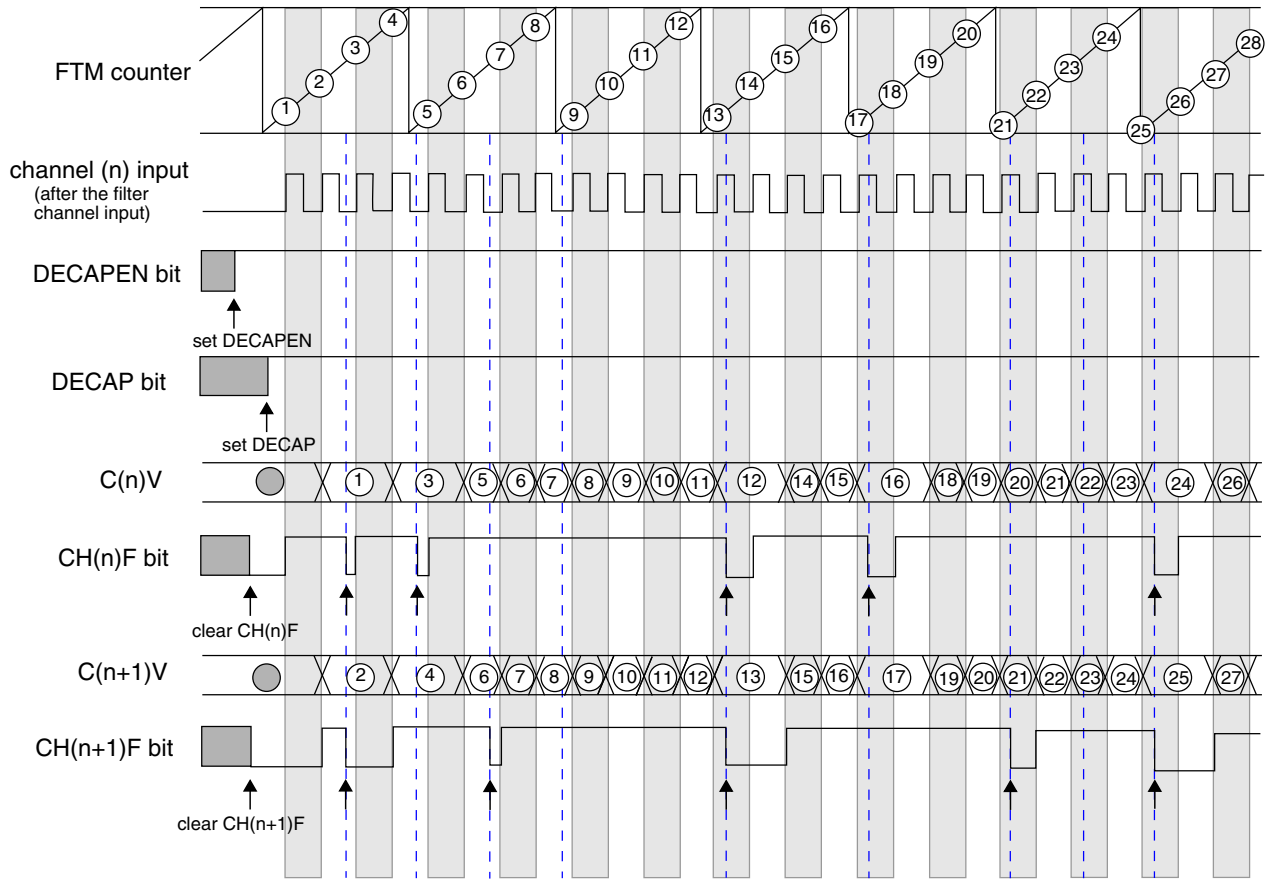
Note

- 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 35-249. Dual Edge Capture – One-Shot mode to measure of the period between two consecutive rising edges**

## Functional description

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 35-250. Dual Edge Capture – Continuous mode to measure of the period between two consecutive rising edges**

### 35.4.24.5 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

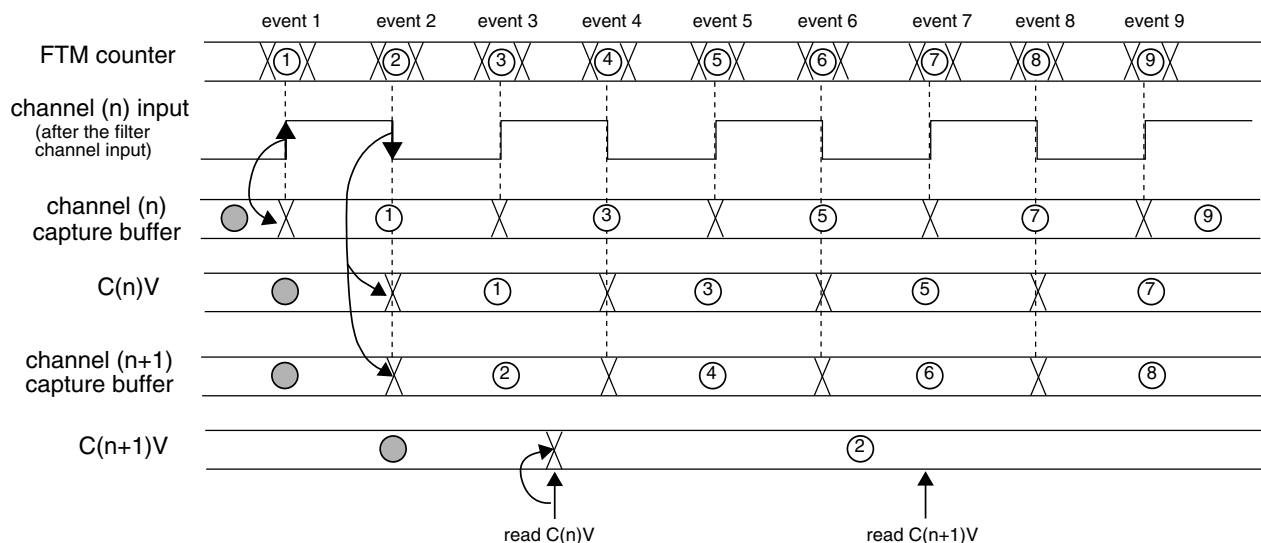


+1) are in Dual Edge Capture – Continuous mode for positive polarity pulse width measurement. Thus, the channel (n) is configured to capture the FTM counter value when there is a rising edge at channel (n) input signal, and channel (n+1) to capture the FTM counter value when there is a falling edge at 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.



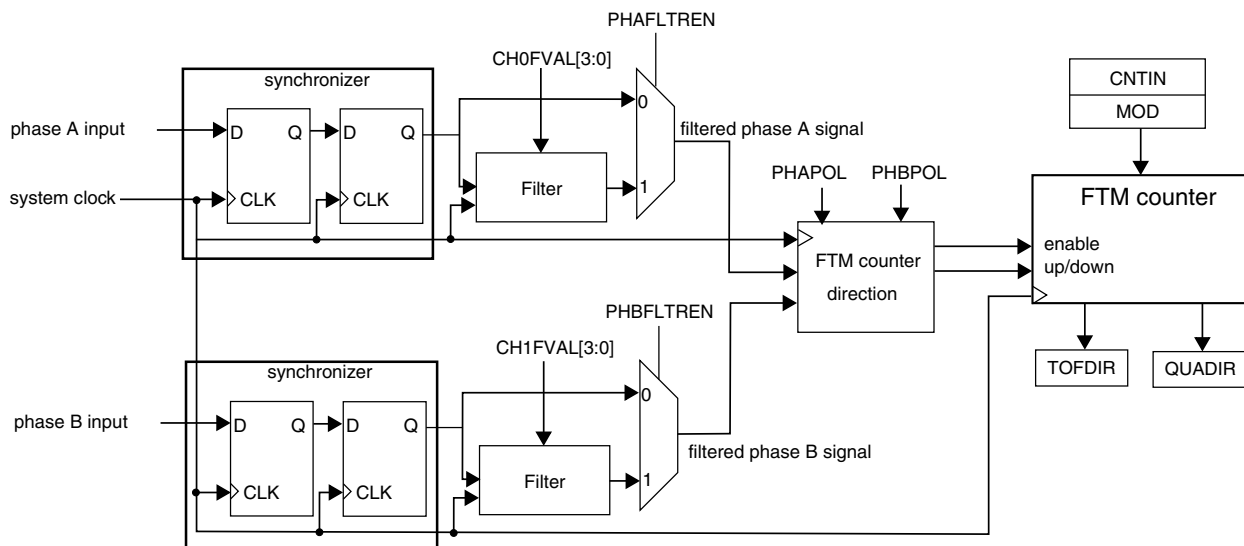
**Figure 35-251. Dual Edge Capture mode read coherency mechanism**

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.

### 35.4.25 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.

## Functional description



**Figure 35-252. 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

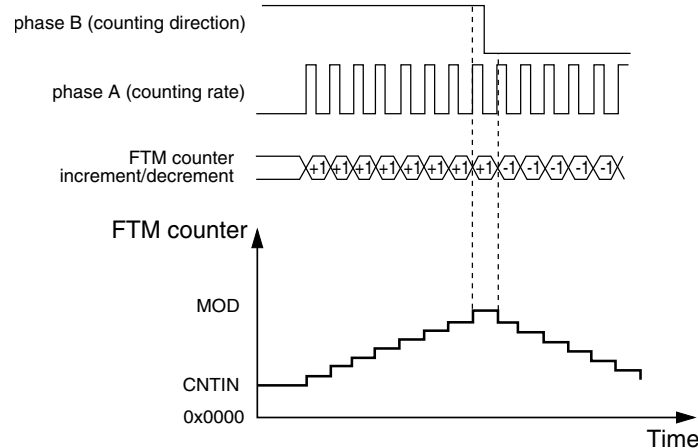
Notice that the FTM counter is clocked by the phase A and B input signals when quadrature decoder mode is selected. Therefore it is expected that the Quadrature Decoder be used only with the FTM channels in input capture or output compare modes.

### 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 35-253. 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.

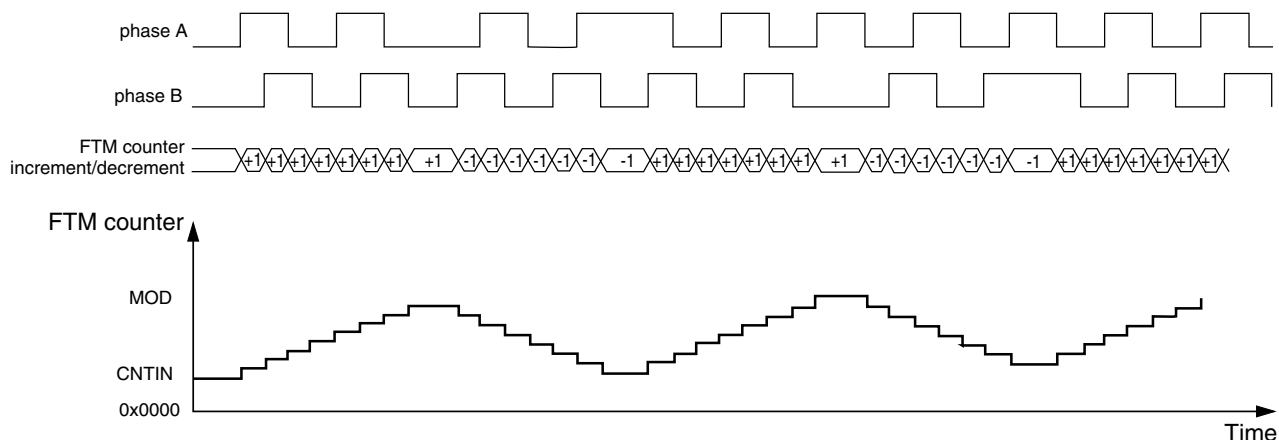
If PHAPOL = 0 and PHBPOL = 0, then the FTM counter increment happens when:

- 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:

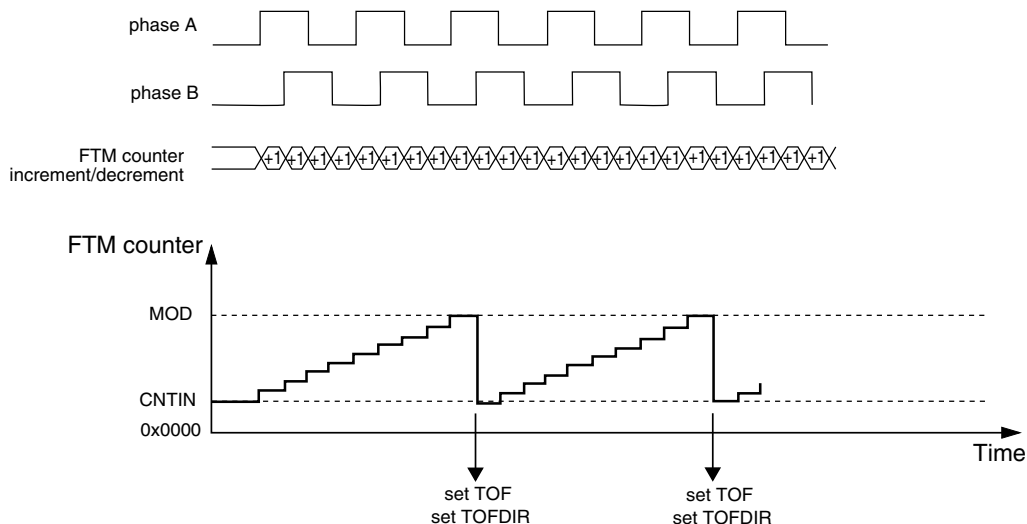
- there is a falling edge at phase A signal and phase B signal is at logic zero;
- there is a falling edge at phase B signal and phase A signal is at logic one;
- there is a rising edge at phase B signal and phase A signal is at logic zero;
- there is a rising edge at phase A signal and phase B signal is at logic one.

## Functional description



**Figure 35-254. 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 35-255. 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.

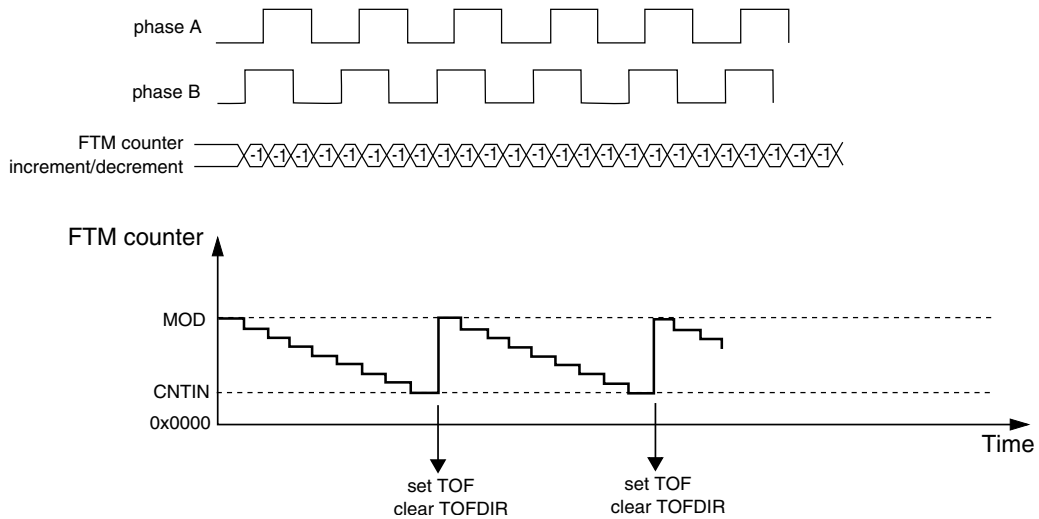


Figure 35-256. FTM counter overflow in down counting for Quadrature Decoder mode

### 35.4.25.1 Quadrature Decoder boundary conditions

The following figures show the FTM counter responding to motor jittering typical in motor position control applications.

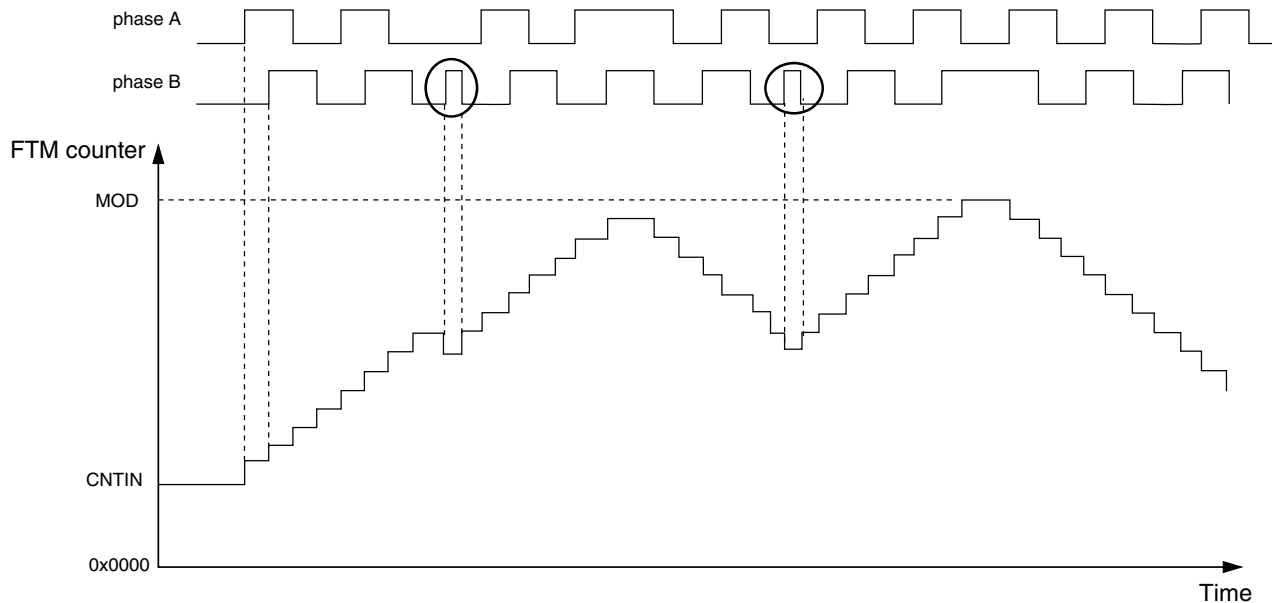
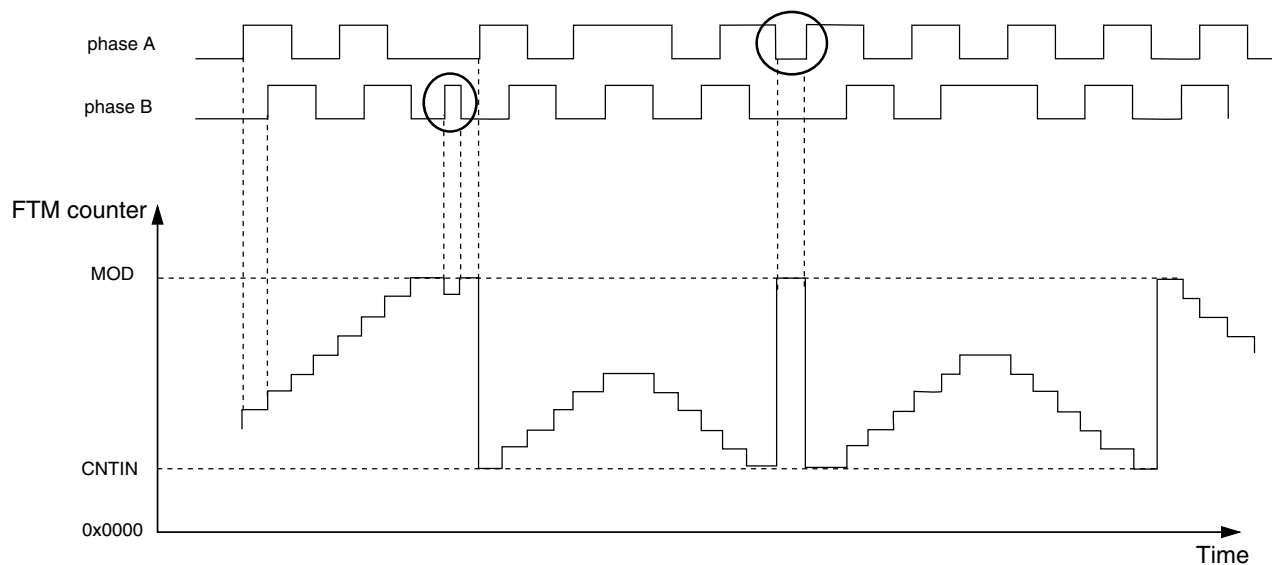


Figure 35-257. Motor position jittering in a mid count value

The following figure shows motor jittering produced by the phase B and A pulses respectively:

## Functional description



**Figure 35-258. 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.

### 35.4.26 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 35-253. 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 35-253. 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.

## 35.4.27 Intermediate load

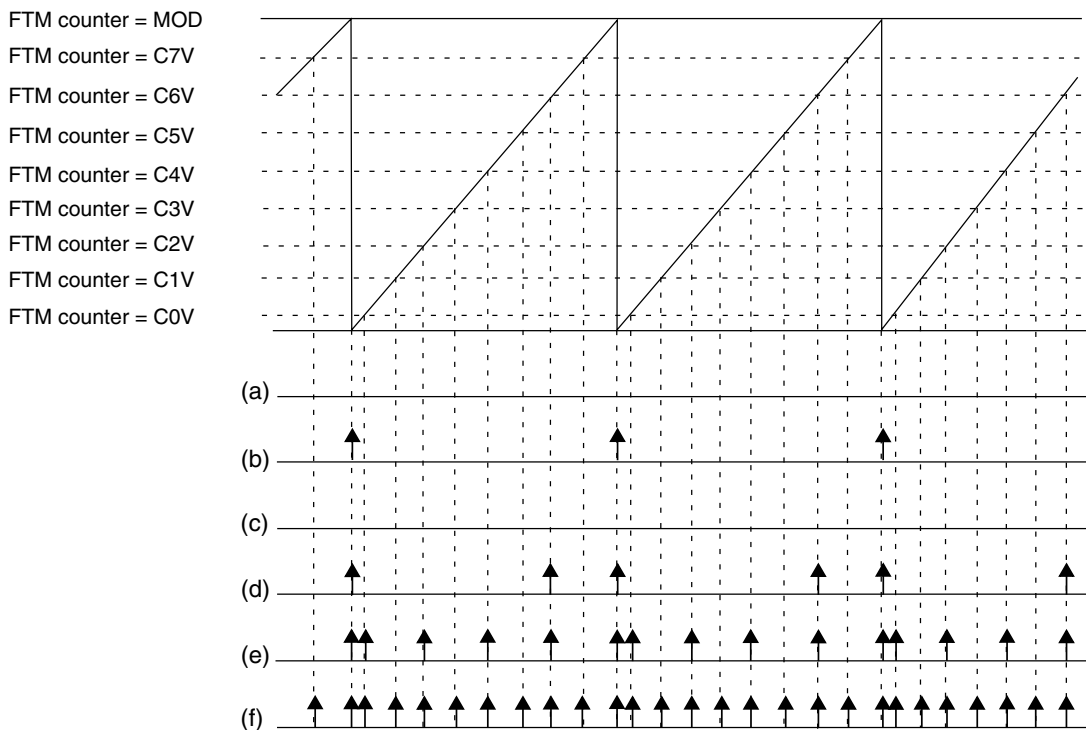
The PWMLOAD register allows to update the MOD, CNTIN, and C(n)V registers with the content of the register buffer at a defined load point. In this case, it is not required to use the PWM synchronization.

There are multiple possible loading points for intermediate load:

**Table 35-254. 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 35-259. 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 35-255. 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 35-255. Conditions for loads occurring at the next enabled loading point (continued)**

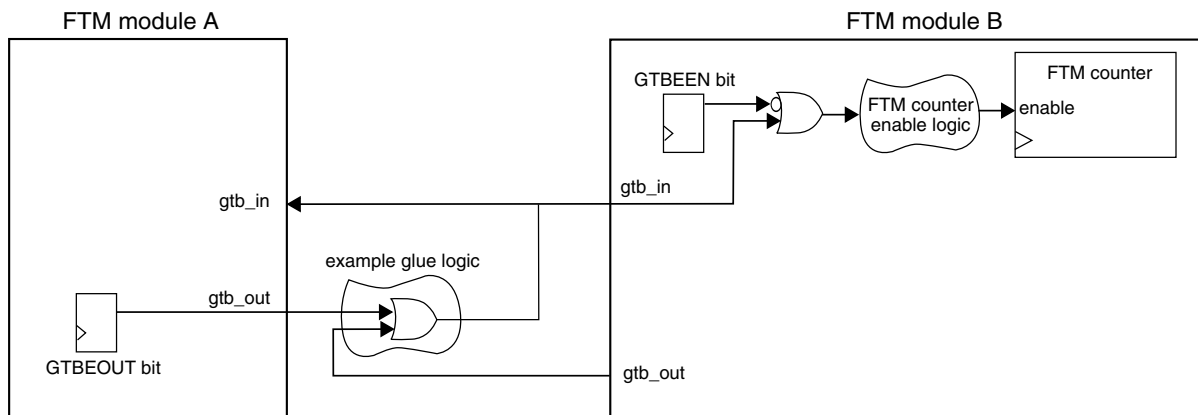
When a new value was written	Then
To the CNTIN register and CNTINC = 1	The CNTIN register is updated with its write buffer value.
To the C(n)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1)	The C(n)V register is updated with its write buffer value.
To the C(n+1)V register 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.

**35.4.28 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 35-260. 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.

#### 35.4.28.1 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.

## 35.5 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).

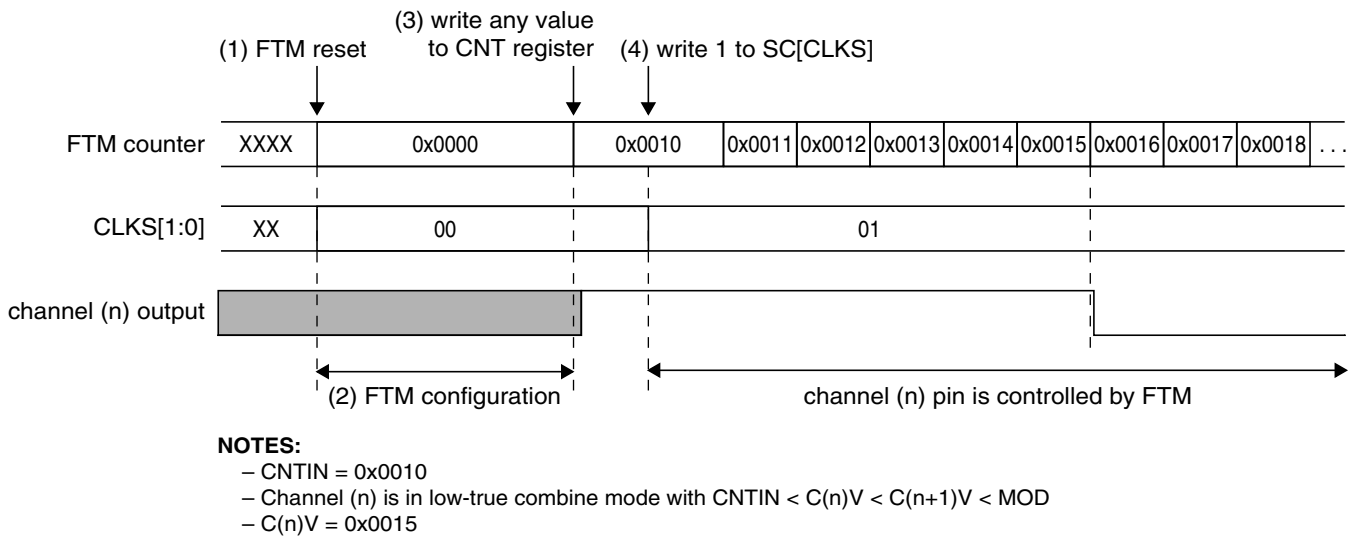
The following figure shows the FTM behavior after the reset. At the reset (item 1), the FTM counter is disabled (see the description of the CLKS field in the Status and Control register), its value is updated to zero and the pins are not controlled by FTM (See the table in the description of CnSC register).

After the reset, the FTM should be configured (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](#)).

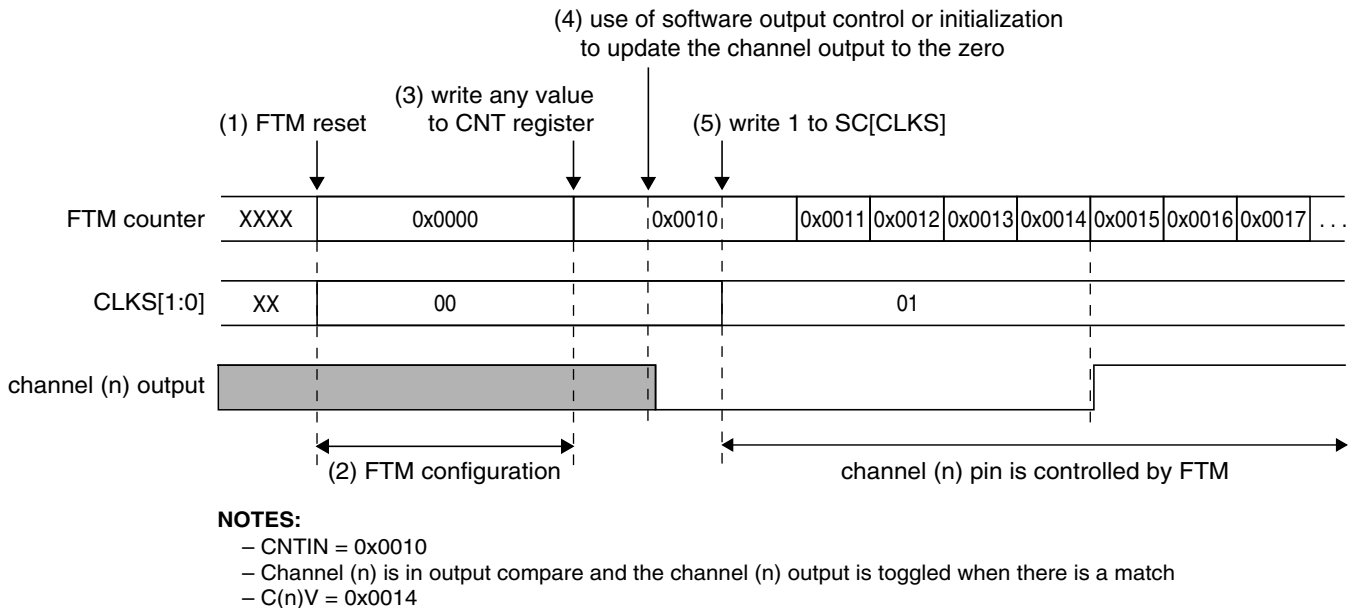
The next step is to select the FTM counter clock by the CLKS[1:0] bits (item 4). It is important to highlight that the pins are only controlled by FTM when CLKS[1:0] bits are different from zero (See the table in the description of CnSC register).

## FTM Interrupts



**Figure 35-261. 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).



**Figure 35-262. FTM behavior after reset when the channel (n) is in Output Compare mode**

## 35.6 FTM Interrupts

### 35.6.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

### 35.6.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

### 35.6.3 Fault Interrupt

The fault interrupt is generated when (FAULTIE = 1) and (FAULTF = 1).

## 35.7 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, HWRBUF = 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 36

# Periodic Interrupt Timer (PIT)

### 36.1 Introduction

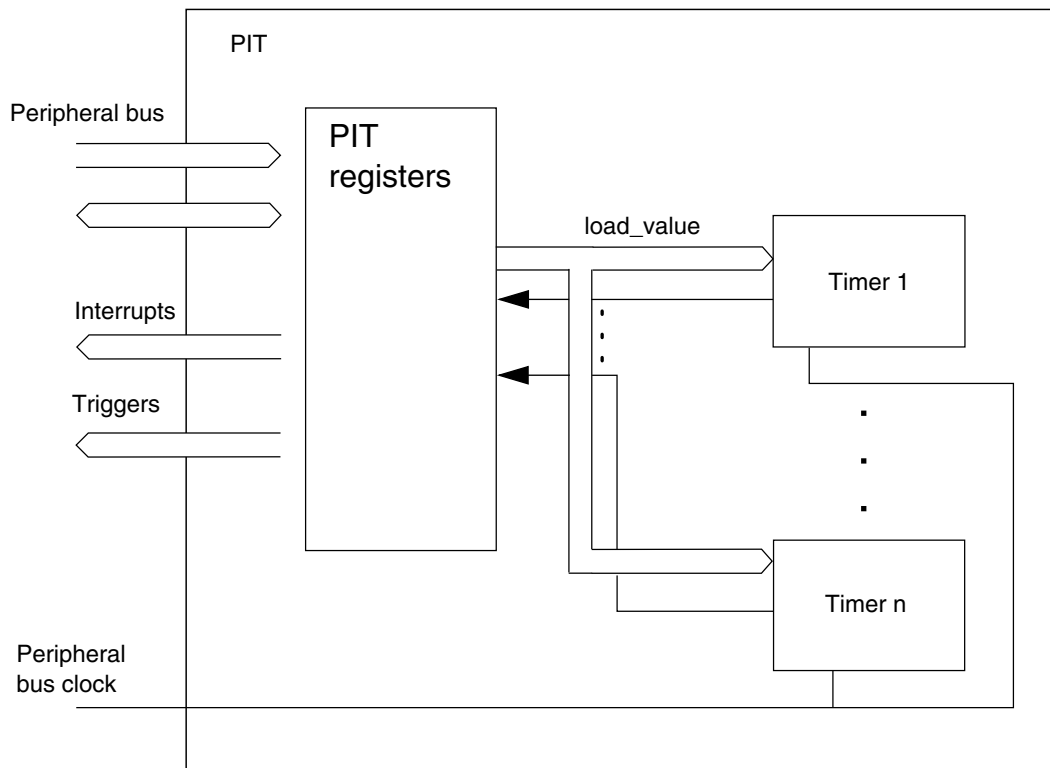
#### NOTE

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

The PIT module is an array of timers that can be used to raise interrupts and trigger DMA channels.

#### 36.1.1 Block diagram

The following figure shows the block diagram of the PIT module.



**Figure 36-1. Block diagram of the PIT**

### NOTE

See the chip-specific PIT information for the number of PIT channels used in this MCU.

## 36.1.2 Features

The main features of this block are:

- Ability of timers to generate DMA trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer

## 36.2 Signal description

The PIT module has no external pins.



## 36.3 Memory map/register description

This section provides a detailed description of all registers accessible in the PIT module.

- Reserved registers will read as 0, writes will have no effect.
- See the chip-specific PIT information for the number of PIT channels used in this MCU.

### PIT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_7000	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0006h	<a href="#">36.3.1/849</a>
4003_7100	Timer Load Value Register (PIT_LDVAL0)	32	R/W	0000_0000h	<a href="#">36.3.2/851</a>
4003_7104	Current Timer Value Register (PIT_CVAL0)	32	R	0000_0000h	<a href="#">36.3.3/851</a>
4003_7108	Timer Control Register (PIT_TCTRL0)	32	R/W	0000_0000h	<a href="#">36.3.4/852</a>
4003_710C	Timer Flag Register (PIT_TFLG0)	32	R/W	0000_0000h	<a href="#">36.3.5/852</a>
4003_7110	Timer Load Value Register (PIT_LDVAL1)	32	R/W	0000_0000h	<a href="#">36.3.2/851</a>
4003_7114	Current Timer Value Register (PIT_CVAL1)	32	R	0000_0000h	<a href="#">36.3.3/851</a>
4003_7118	Timer Control Register (PIT_TCTRL1)	32	R/W	0000_0000h	<a href="#">36.3.4/852</a>
4003_711C	Timer Flag Register (PIT_TFLG1)	32	R/W	0000_0000h	<a href="#">36.3.5/852</a>
4003_7120	Timer Load Value Register (PIT_LDVAL2)	32	R/W	0000_0000h	<a href="#">36.3.2/851</a>
4003_7124	Current Timer Value Register (PIT_CVAL2)	32	R	0000_0000h	<a href="#">36.3.3/851</a>
4003_7128	Timer Control Register (PIT_TCTRL2)	32	R/W	0000_0000h	<a href="#">36.3.4/852</a>
4003_712C	Timer Flag Register (PIT_TFLG2)	32	R/W	0000_0000h	<a href="#">36.3.5/852</a>
4003_7130	Timer Load Value Register (PIT_LDVAL3)	32	R/W	0000_0000h	<a href="#">36.3.2/851</a>
4003_7134	Current Timer Value Register (PIT_CVAL3)	32	R	0000_0000h	<a href="#">36.3.3/851</a>
4003_7138	Timer Control Register (PIT_TCTRL3)	32	R/W	0000_0000h	<a href="#">36.3.4/852</a>
4003_713C	Timer Flag Register (PIT_TFLG3)	32	R/W	0000_0000h	<a href="#">36.3.5/852</a>

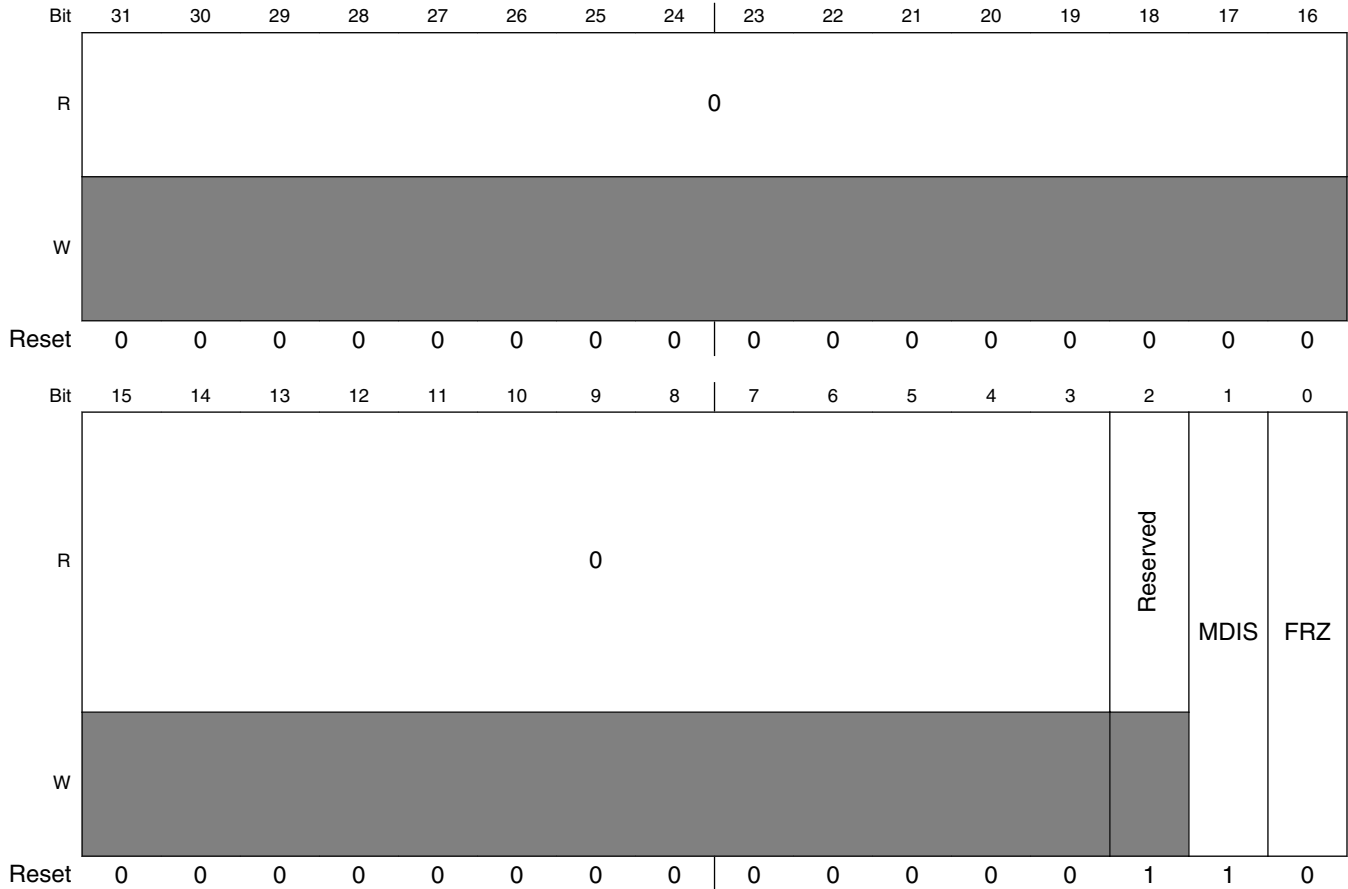
### 36.3.1 PIT Module Control Register (PIT\_MCR)

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

Access: User read/write

## Memory map/register description

Address: 4003\_7000h base + 0h offset = 4003\_7000h



### PIT\_MCR field descriptions

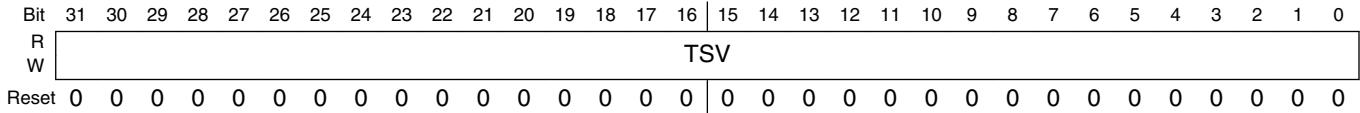
Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved.
1 MDIS	Module Disable - (PIT section)  Disables the standard timers. This field must be enabled before any other setup is done.  0 Clock for standard PIT timers is enabled. 1 Clock for standard PIT timers is disabled.
0 FRZ	Freeze  Allows the timers to be stopped when the device enters the Debug mode.  0 Timers continue to run in Debug mode. 1 Timers are stopped in Debug mode.

### 36.3.2 Timer Load Value Register (PIT\_LDVAL<sub>n</sub>)

These registers select the timeout period for the timer interrupts.

Access: User read/write

Address: 4003\_7000h base + 100h offset + (16d × i), where i=0d to 3d



#### PIT\_LDVAL<sub>n</sub> field descriptions

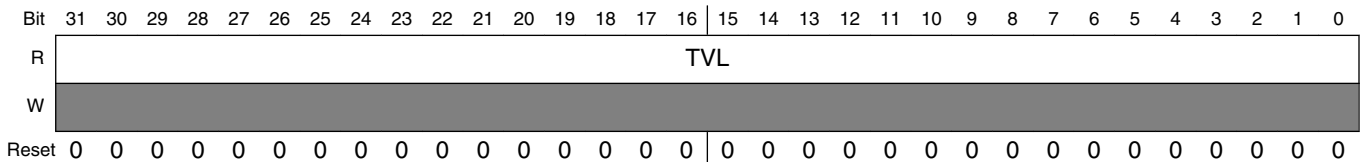
Field	Description
31–0 TSV	<p>Timer Start Value</p> <p>Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.</p>

### 36.3.3 Current Timer Value Register (PIT\_CVAL<sub>n</sub>)

These registers indicate the current timer position.

Access: User read only

Address: 4003\_7000h base + 104h offset + (16d × i), where i=0d to 3d



#### PIT\_CVAL<sub>n</sub> field descriptions

Field	Description
31–0 TVL	<p>Current Timer Value</p> <p>Represents the current timer value, if the timer is enabled.</p> <p><b>NOTE:</b></p> <ul style="list-style-type: none"> <li>If the timer is disabled, do not use this field as its value is unreliable.</li> <li>The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.</li> </ul>

### 36.3.4 Timer Control Register (PIT\_TCTRLn)

These registers contain the control bits for each timer.

Access: User read/write

Address: 4003\_7000h base + 108h offset + (16d × i), where i=0d to 3d

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													CHN	TIE	TEN	
W																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

#### PIT\_TCTRLn field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CHN	Chain Mode When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer. For example, for Channel 2, if this field is set, Timer 2 is chained to Timer 1.
1 TIE	Timer Interrupt Enable When an interrupt is pending, or, TFLGn[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TFLGn[TIF] must be cleared first. 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.
0 TEN	Timer Enable Enables or disables the timer. 0 Timer n is disabled. 1 Timer n is enabled.

### 36.3.5 Timer Flag Register (PIT\_TFLGn)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 4003\_7000h base + 10Ch offset + (16d × i), where i=0d to 3d

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															TIF	
W																w1c	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

### PIT\_TFLGn field descriptions

Field	Description
31–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 TIF	<p>Timer Interrupt Flag</p> <p>Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF causes an interrupt request.</p> <p>0 Timeout has not yet occurred. 1 Timeout has occurred.</p>

## 36.4 Functional description

This section provides the functional description of the module.

### 36.4.1 General operation

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses and interrupts. Each interrupt is available on a separate interrupt line.

#### 36.4.1.1 Timers

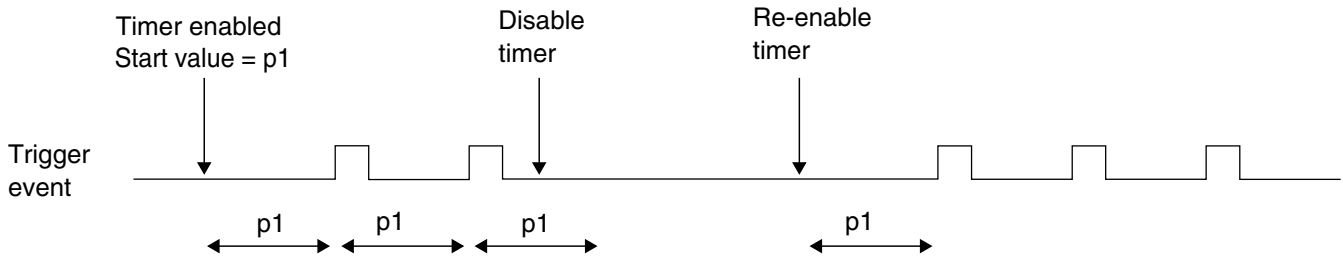
The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked by setting TCTRLn[TIE]. A new interrupt can be generated only after the previous one is cleared.

## Functional description

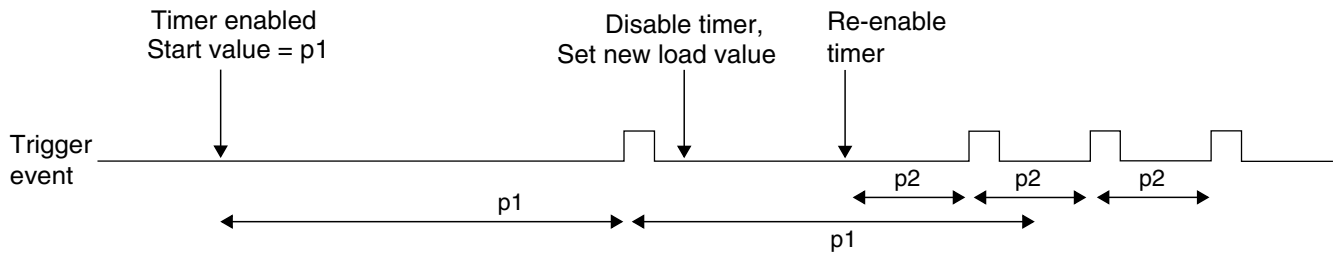
If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN]. See the following figure.



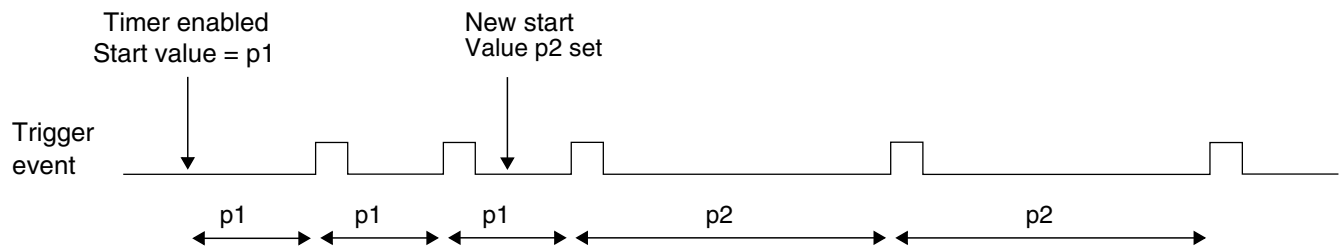
**Figure 36-23. Stopping and starting a timer**

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value, and then enabling the timer again. See the following figure.



**Figure 36-24. Modifying running timer period**

It is also possible to change the counter period without restarting the timer by writing LDVAL with the new load value. This value will then be loaded after the next trigger event. See the following figure.



**Figure 36-25. Dynamically setting a new load value**

### 36.4.1.2 Debug mode

In Debug mode, the timers will be frozen based on MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system, for example, the timer values, and then continue the operation.

### 36.4.2 Interrupts

All the timers support interrupt generation. See the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting TCTRLn[TIE]. TFLGn[TIF] are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to the corresponding TFLGn[TIF].

### 36.4.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

## 36.5 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

The PIT module must be activated by writing a 0 to MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns. Timer 1 needs to trigger every  $5.12 \text{ ms} / 20 \text{ ns} = 256,000$  cycles and Timer 3 every  $30 \text{ ms} / 20 \text{ ns} = 1,500,000$  cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) - 1

## Example configuration for chained timers

This means LDVAL1 and LDVAL3 must be written with 0x0003E7FF and 0x0016E35F respectively.

The interrupt for Timer 1 is enabled by setting TCTRL1[TIE]. The timer is started by writing 1 to TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore, Timer 3 is started by writing a 1 to TCTRL3[TEN]. TCTRL3[TIE] stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;

// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

## 36.6 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 minute.

The PIT module needs to be activated by writing a 0 to MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So, Timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to Timer 1 and programmed to trigger 10 times.

The value for the LDVAL register trigger is calculated as number of cycles-1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for Timer 2 is enabled by setting TCTRL2[TIE], the Chain mode is activated by setting TCTRL2[CHN], and the timer is started by writing a 1 to TCTRL2[TEN]. TCTRL1[TEN] needs to be set, and TCTRL1[CHN] and TCTRL1[TIE] are cleared.



The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;

// Timer 2
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2

// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```



# 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
LPTMR_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	<a href="#">37.3.1/861</a>
4004_0004	Low Power Timer Prescale Register (LPTMR0_PSR)	32	R/W	0000_0000h	<a href="#">37.3.2/862</a>
4004_0008	Low Power Timer Compare Register (LPTMR0_CMR)	32	R/W	0000_0000h	<a href="#">37.3.3/864</a>
4004_000C	Low Power Timer Counter Register (LPTMR0_CNR)	32	R	0000_0000h	<a href="#">37.3.4/864</a>

## 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	<p>Prescale Value</p> <p>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.</p> <p>0000 Prescaler divides the prescaler clock by 2; glitch filter does not support this configuration.</p> <p>0001 Prescaler divides the prescaler clock by 4; glitch filter recognizes change on input pin after 2 rising clock edges.</p> <p>0010 Prescaler divides the prescaler clock by 8; glitch filter recognizes change on input pin after 4 rising clock edges.</p> <p>0011 Prescaler divides the prescaler clock by 16; glitch filter recognizes change on input pin after 8 rising clock edges.</p> <p>0100 Prescaler divides the prescaler clock by 32; glitch filter recognizes change on input pin after 16 rising clock edges.</p> <p>0101 Prescaler divides the prescaler clock by 64; glitch filter recognizes change on input pin after 32 rising clock edges.</p> <p>0110 Prescaler divides the prescaler clock by 128; glitch filter recognizes change on input pin after 64 rising clock edges.</p> <p>0111 Prescaler divides the prescaler clock by 256; glitch filter recognizes change on input pin after 128 rising clock edges.</p> <p>1000 Prescaler divides the prescaler clock by 512; glitch filter recognizes change on input pin after 256 rising clock edges.</p> <p>1001 Prescaler divides the prescaler clock by 1024; glitch filter recognizes change on input pin after 512 rising clock edges.</p> <p>1010 Prescaler divides the prescaler clock by 2048; glitch filter recognizes change on input pin after 1024 rising clock edges.</p> <p>1011 Prescaler divides the prescaler clock by 4096; glitch filter recognizes change on input pin after 2048 rising clock edges.</p> <p>1100 Prescaler divides the prescaler clock by 8192; glitch filter recognizes change on input pin after 4096 rising clock edges.</p> <p>1101 Prescaler divides the prescaler clock by 16,384; glitch filter recognizes change on input pin after 8192 rising clock edges.</p> <p>1110 Prescaler divides the prescaler clock by 32,768; glitch filter recognizes change on input pin after 16,384 rising clock edges.</p> <p>1111 Prescaler divides the prescaler clock by 65,536; glitch filter recognizes change on input pin after 32,768 rising clock edges.</p>
2 PBYP	<p>Prescaler Bypass</p> <p>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.</p> <p>0 Prescaler/glitch filter is enabled.</p> <p>1 Prescaler/glitch filter is bypassed.</p>
1–0 PCS	<p>Prescaler Clock Select</p> <p>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.</p> <p><b>NOTE:</b> See the chip configuration details for information on the connections to these inputs.</p>

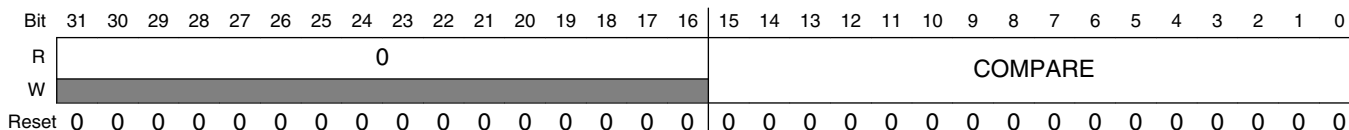
*Table continues on the next page...*

**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\_CMCR)**

Address: 4004\_0000h base + 8h offset = 4004\_0008h

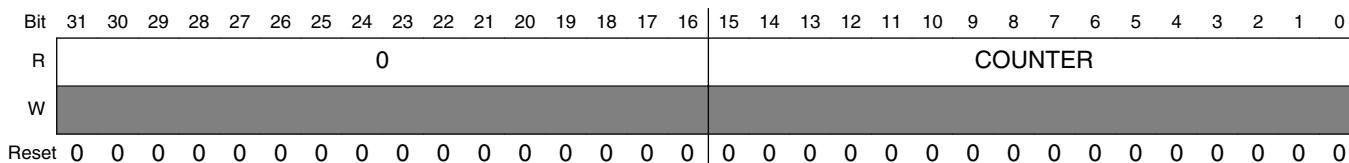


**LPTMRx\_CMCR field descriptions**

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–0 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)**

Address: 4004\_0000h base + Ch offset = 4004\_000Ch



**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.
15–0 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 may 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

## Serial Peripheral Interface (SPI)

### 38.1 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 an MCU and an external peripheral device.

#### 38.1.1 Block Diagram

The block diagram of this module is as follows:

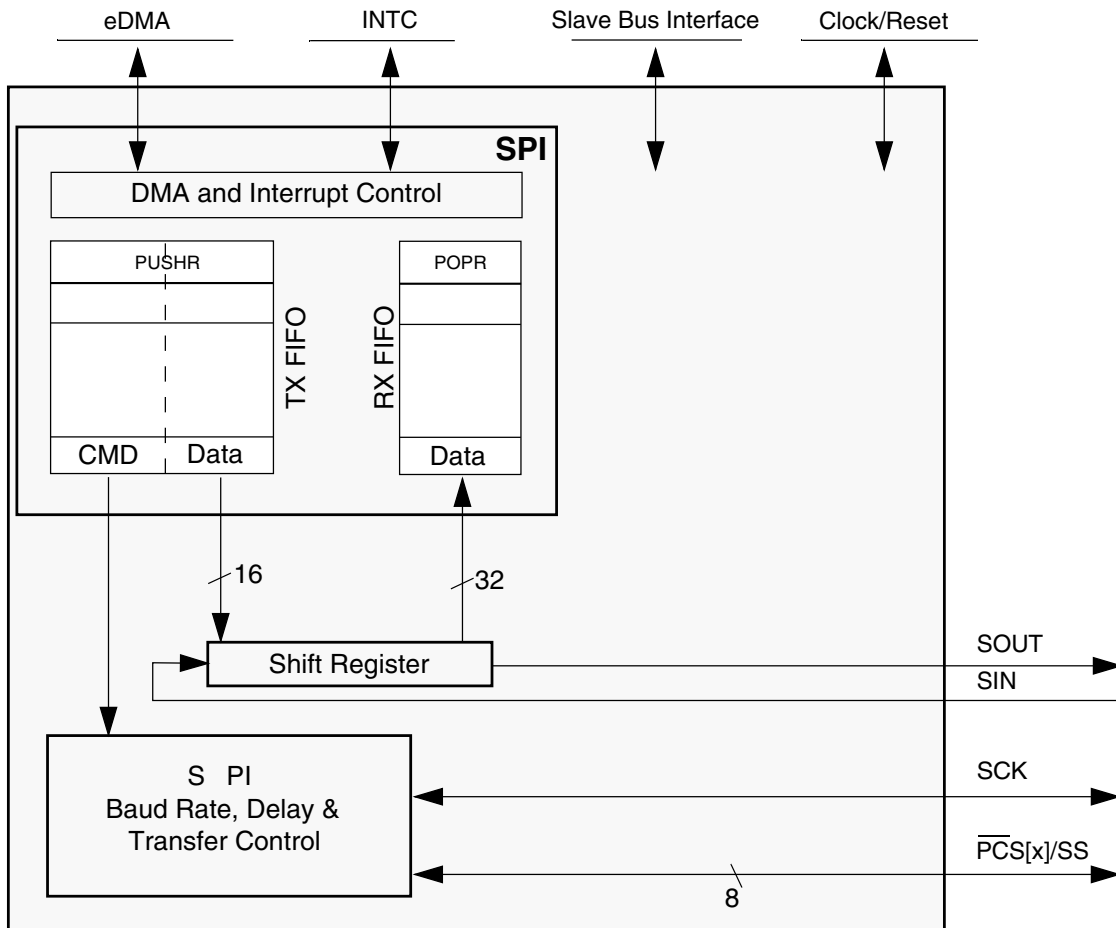


Figure 38-1. SPI Block Diagram

### 38.1.2 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
- 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 bits
    - SPI frames longer than 16 bits can be supported using the continuous selection format.
  - Continuously held chip select capability
- 6 peripheral chip selects (PCSEs), expandable to 64 with external demultiplexer
- Deglitching support for up to 32 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

### 38.1.3 Interface configurations

#### 38.1.3.1 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.

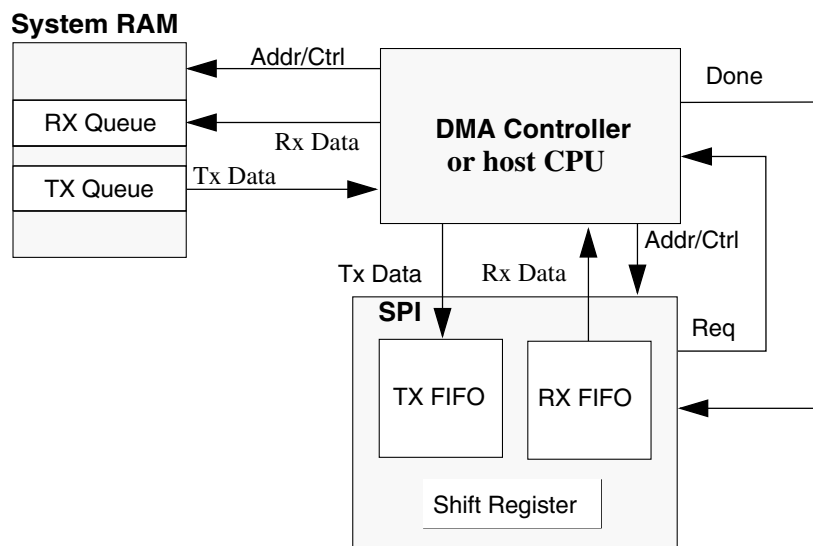


Figure 38-2. SPI with queues and DMA

#### 38.1.4 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
- MCU-specific modes:
  - External Stop mode
  - Debug mode

The module enters module-specific modes when the host writes a module register. The MCU-specific modes are controlled by signals external to the module. The MCU-specific modes are modes that an MCU may enter in parallel to the block-specific modes.

#### **38.1.4.1 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]

#### **38.1.4.2 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]/ $\overline{SS}$  signals are configured as inputs and driven by an SPI bus master.

#### **38.1.4.3 Module Disable Mode**

The Module Disable mode can be used for MCU power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

### 38.1.4.4 External Stop Mode

External Stop mode is used for MCU 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.

### 38.1.4.5 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 MCU is in debug mode.
- If the bit is cleared, the MCU debug mode has no effect on the module.

## 38.2 Module signal descriptions

This table describes the signals on the boundary of the module that may connect off chip (in alphabetical order).

**Table 38-1. 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
PCS4	Peripheral Chip Select 4	(Unused)	O
PCS5/ $\overline{PCSS}$	Peripheral Chip Select 5 /Peripheral Chip Select Strobe	(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

### 38.2.1 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.

### 38.2.2 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

### 38.2.3 PCS4—Peripheral Chip Select 4

Master mode: Peripheral Chip Select 4 (O)—Selects an SPI slave to receive data transmitted by the module.

Slave mode: Unused

### 38.2.4 PCS5/ $\overline{\text{PCSS}}$ —Peripheral Chip Select 5/Peripheral Chip Select Strobe

Master mode:

- Peripheral Chip Select 5 (O)—Used only when the peripheral-chip-select strobe is disabled (MCR[PCSSE]). Selects an SPI slave to receive data transmitted by the module.
- Peripheral Chip Select Strobe (O)—Used only when the peripheral-chip-select strobe is enabled (MCR[PCSSE]). Strokes an off-module peripheral-chip-select demultiplexer, which decodes the module's PCS signals other than PCS5, preventing glitches on the demultiplexer outputs.

Slave mode: Unused

### 38.2.5 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.

### 38.2.6 SIN—Serial Input

Master mode: Serial Input (I)—Receives serial data.

Slave mode: Serial Input (I)—Receives serial data.

**NOTE**

Serial Data Out output buffers are controlled through SIU (or SIUL) and cannot be controlled through the module.

**38.2.7 SOUT—Serial Output**

Master mode: Serial Output (O)—Transmits serial data.

Slave mode: Serial Output (O)—Transmits serial data.

**NOTE**

Serial Data Out output buffers are controlled through SIU (or SIUL) and cannot be controlled through the module.

**38.3 Memory Map/Register Definition**

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the POPR 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	<a href="#">38.3.1/878</a>
4002_C008	Transfer Count Register (SPI0_TCR)	32	R/W	0000_0000h	<a href="#">38.3.2/881</a>
4002_C00C	Clock and Transfer Attributes Register (In Master Mode) (SPI0_CTAR0)	32	R/W	7800_0000h	<a href="#">38.3.3/881</a>
4002_C00C	Clock and Transfer Attributes Register (In Slave Mode) (SPI0_CTAR0_SLAVE)	32	R/W	7800_0000h	<a href="#">38.3.4/886</a>
4002_C010	Clock and Transfer Attributes Register (In Master Mode) (SPI0_CTAR1)	32	R/W	7800_0000h	<a href="#">38.3.3/881</a>
4002_C02C	Status Register (SPI0_SR)	32	R/W	<a href="#">See section</a>	<a href="#">38.3.5/888</a>
4002_C030	DMA/Interrupt Request Select and Enable Register (SPI0_RSER)	32	R/W	0000_0000h	<a href="#">38.3.6/891</a>
4002_C034	PUSH TX FIFO Register In Master Mode (SPI0_PUSHR)	32	R/W	0000_0000h	<a href="#">38.3.7/893</a>
4002_C034	PUSH TX FIFO Register In Slave Mode (SPI0_PUSHR_SLAVE)	32	R/W	0000_0000h	<a href="#">38.3.8/894</a>
4002_C038	POP RX FIFO Register (SPI0_POPR)	32	R	0000_0000h	<a href="#">38.3.9/895</a>
4002_C03C	Transmit FIFO Registers (SPI0_TXFR0)	32	R	0000_0000h	<a href="#">38.3.10/896</a>
4002_C040	Transmit FIFO Registers (SPI0_TXFR1)	32	R	0000_0000h	<a href="#">38.3.10/896</a>

*Table continues on the next page...*

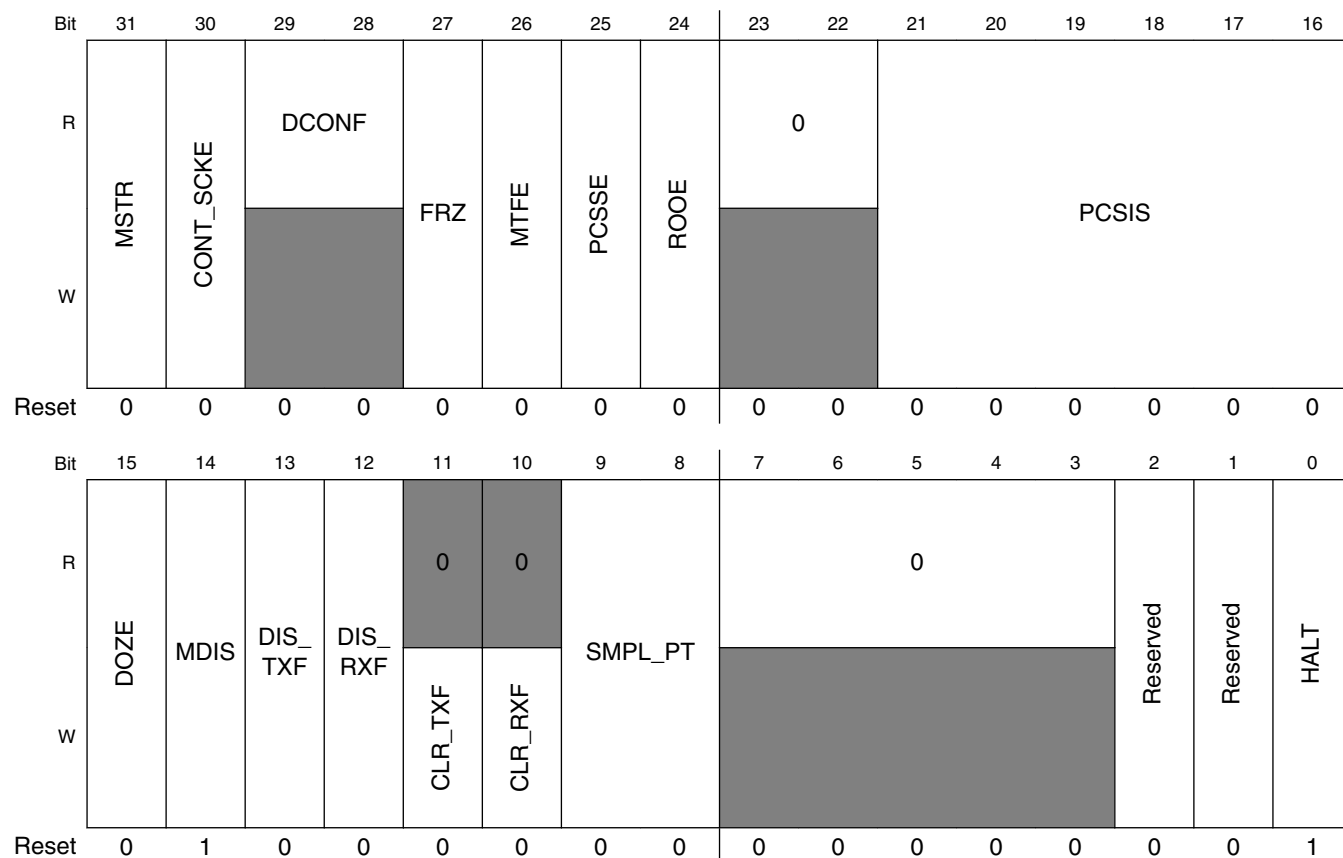
## SPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4002_C044	Transmit FIFO Registers (SPI0_TXFR2)	32	R	0000_0000h	<a href="#">38.3.10/896</a>
4002_C048	Transmit FIFO Registers (SPI0_TXFR3)	32	R	0000_0000h	<a href="#">38.3.10/896</a>
4002_C07C	Receive FIFO Registers (SPI0_RXFR0)	32	R	0000_0000h	<a href="#">38.3.11/896</a>
4002_C080	Receive FIFO Registers (SPI0_RXFR1)	32	R	0000_0000h	<a href="#">38.3.11/896</a>
4002_C084	Receive FIFO Registers (SPI0_RXFR2)	32	R	0000_0000h	<a href="#">38.3.11/896</a>
4002_C088	Receive FIFO Registers (SPI0_RXFR3)	32	R	0000_0000h	<a href="#">38.3.11/896</a>

### 38.3.1 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.

Table continues on the next page...

## SPIx\_MCR field descriptions (continued)

Field	Description
	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 Timing Format Enable  Enables a modified transfer format to be used.  0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
25 PCSSE	Peripheral Chip Select Strobe Enable  Enables the PCS5/ $\overline{PCSS}$ to operate as a PCS Strobe output signal.  0 PCS5/ $\overline{PCSS}$ is used as the Peripheral Chip Select[5] signal. 1 PCS5/ $\overline{PCSS}$ is used as an active-low PCS Strobe signal.
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.
23–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–16 PCSiS	Peripheral Chip Select x Inactive State  Determines the inactive state of PCSx.  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.

Table continues on the next page...

## SPIx\_MCR field descriptions (continued)

Field	Description
14 MDIS	<p>Module Disable</p> <p>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 0. 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.</p> <p>0 Enables the module clocks. 1 Allows external logic to disable the module clocks.</p>
13 DIS_TXF	<p>Disable Transmit FIFO</p> <p>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.</p> <p>0 TX FIFO is enabled. 1 TX FIFO is disabled.</p>
12 DIS_RXF	<p>Disable Receive FIFO</p> <p>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.</p> <p>0 RX FIFO is enabled. 1 RX FIFO is disabled.</p>
11 CLR_TXF	<p>Clear TX FIFO</p> <p>Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero.</p> <p>0 Do not clear the TX FIFO counter. 1 Clear the TX FIFO counter.</p>
10 CLR_RXF	<p>CLR_RXF</p> <p>Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero.</p> <p>0 Do not clear the RX FIFO counter. 1 Clear the RX FIFO counter.</p>
9–8 SMPL_PT	<p>Sample Point</p> <p>Controls when the module master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTARn[CPHA] is 0.</p> <p>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</p>
7–3 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
2 Reserved	<p>This field is reserved.</p>

Table continues on the next page...



**SPIx\_MCR field descriptions (continued)**

Field	Description
1 Reserved	This field is reserved.
0 HALT	<p>Halt</p> <p>The HALT bit starts and stops frame transfers. See <a href="#">Start and Stop of Module transfers</a></p> <p>0 Start transfers. 1 Stop transfers.</p>

**38.3.2 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	<p>SPI Transfer Counter</p> <p>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.</p>
15–0 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>

**38.3.3 Clock and Transfer Attributes Register (In Master Mode) (SPIx\_CTARn)**

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

## Memory Map/Register Definition

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 an 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																
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																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### SPIx\_CTARn field descriptions

Field	Description																																								
31 DBR	<p>Double Baud Rate</p> <p>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.</p> <p style="text-align: center;"><b>Table 38-40. SPI SCK Duty Cycle</b></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>DBR</th> <th>CPHA</th> <th>PBR</th> <th>SCK Duty Cycle</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>any</td> <td>any</td> <td>50/50</td> </tr> <tr> <td>1</td> <td>0</td> <td>00</td> <td>50/50</td> </tr> <tr> <td>1</td> <td>0</td> <td>01</td> <td>33/66</td> </tr> <tr> <td>1</td> <td>0</td> <td>10</td> <td>40/60</td> </tr> <tr> <td>1</td> <td>0</td> <td>11</td> <td>43/57</td> </tr> <tr> <td>1</td> <td>1</td> <td>00</td> <td>50/50</td> </tr> <tr> <td>1</td> <td>1</td> <td>01</td> <td>66/33</td> </tr> <tr> <td>1</td> <td>1</td> <td>10</td> <td>60/40</td> </tr> <tr> <td>1</td> <td>1</td> <td>11</td> <td>57/43</td> </tr> </tbody> </table>	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
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																																						

Table continues on the next page...

## SPIx\_CTARn field descriptions (continued)

Field	Description
	<p>0 The baud rate is computed normally with a 50/50 duty cycle.</p> <p>1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.</p>
30–27 FMSZ	<p>Frame Size</p> <p>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.</p>
26 CPOL	<p>Clock Polarity</p> <p>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.</p> <p><b>NOTE:</b> In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed.</p> <p>0 The inactive state value of SCK is low.</p> <p>1 The inactive state value of SCK is high.</p>
25 CPHA	<p>Clock Phase</p> <p>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.</p> <p>0 Data is captured on the leading edge of SCK and changed on the following edge.</p> <p>1 Data is changed on the leading edge of SCK and captured on the following edge.</p>
24 LSBFE	<p>LSB First</p> <p>Specifies whether the LSB or MSB of the frame is transferred first.</p> <p>0 Data is transferred MSB first.</p> <p>1 Data is transferred LSB first.</p>
23–22 PCSSCK	<p>PCS to SCK Delay Prescaler</p> <p>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 <a href="#">PCS to SCK Delay (<math>t_{csc}</math>)</a> for more details.</p> <p>00 PCS to SCK Prescaler value is 1.</p> <p>01 PCS to SCK Prescaler value is 3.</p> <p>10 PCS to SCK Prescaler value is 5.</p> <p>11 PCS to SCK Prescaler value is 7.</p>
21–20 PASC	<p>After SCK Delay Prescaler</p> <p>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 <a href="#">After SCK Delay (<math>t_{asc}</math>)</a> for more details.</p> <p>00 Delay after Transfer Prescaler value is 1.</p> <p>01 Delay after Transfer Prescaler value is 3.</p>

Table continues on the next page...

## SPIx\_CTARn field descriptions (continued)

Field	Description																												
	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 <a href="#">Delay after Transfer (t<sub>DT</sub>)</a> 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.																												
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.  <p style="text-align: center;"><b>Table 38-39. Delay Scaler Encoding</b></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Field Value</th> <th>Delay Scaler Value</th> </tr> </thead> <tbody> <tr><td>0000</td><td>2</td></tr> <tr><td>0001</td><td>4</td></tr> <tr><td>0010</td><td>8</td></tr> <tr><td>0011</td><td>16</td></tr> <tr><td>0100</td><td>32</td></tr> <tr><td>0101</td><td>64</td></tr> <tr><td>0110</td><td>128</td></tr> <tr><td>0111</td><td>256</td></tr> <tr><td>1000</td><td>512</td></tr> <tr><td>1001</td><td>1024</td></tr> <tr><td>1010</td><td>2048</td></tr> <tr><td>1011</td><td>4096</td></tr> <tr><td>1100</td><td>8192</td></tr> </tbody> </table>	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
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																												

*Table continues on the next page...*

## SPIx\_CTARn field descriptions (continued)

Field	Description																		
	<p align="center"><b>Table 38-39. Delay Scaler Encoding (continued)</b></p> <table border="1"> <thead> <tr> <th>Field Value</th> <th>Delay Scaler Value</th> </tr> </thead> <tbody> <tr> <td>1101</td> <td>16384</td> </tr> <tr> <td>1110</td> <td>32768</td> </tr> <tr> <td>1111</td> <td>65536</td> </tr> </tbody> </table> <p>Refer <a href="#">PCS to SCK Delay (t<sub>CSC</sub>)</a> for more details.</p>	Field Value	Delay Scaler Value	1101	16384	1110	32768	1111	65536										
Field Value	Delay Scaler Value																		
1101	16384																		
1110	32768																		
1111	65536																		
11–8 ASC	<p>After SCK Delay Scaler</p> <p>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:</p> $t_{ASC} = (1/f_P) \times PASC \times ASC$ <p>See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer <a href="#">After SCK Delay (t<sub>ASC</sub>)</a> for more details.</p>																		
7–4 DT	<p>Delay After Transfer Scaler</p> <p>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.</p> <p>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:</p> $t_{DT} = (1/f_P) \times PDT \times DT$ <p>See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values.</p>																		
3–0 BR	<p>Baud Rate Scaler</p> <p>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:</p> $\text{SCK baud rate} = (f_P / \text{PBR}) \times [(1 + \text{DBR}) / \text{BR}]$ <p>The following table lists the baud rate scaler values.</p> <p align="center"><b>Table 38-38. Baud Rate Scaler</b></p> <table border="1"> <thead> <tr> <th>CTARn[BR]</th> <th>Baud Rate Scaler Value</th> </tr> </thead> <tbody> <tr> <td>0000</td> <td>2</td> </tr> <tr> <td>0001</td> <td>4</td> </tr> <tr> <td>0010</td> <td>6</td> </tr> <tr> <td>0011</td> <td>8</td> </tr> <tr> <td>0100</td> <td>16</td> </tr> <tr> <td>0101</td> <td>32</td> </tr> <tr> <td>0110</td> <td>64</td> </tr> <tr> <td>0111</td> <td>128</td> </tr> </tbody> </table>	CTARn[BR]	Baud Rate Scaler Value	0000	2	0001	4	0010	6	0011	8	0100	16	0101	32	0110	64	0111	128
CTARn[BR]	Baud Rate Scaler Value																		
0000	2																		
0001	4																		
0010	6																		
0011	8																		
0100	16																		
0101	32																		
0110	64																		
0111	128																		

Table continues on the next page...

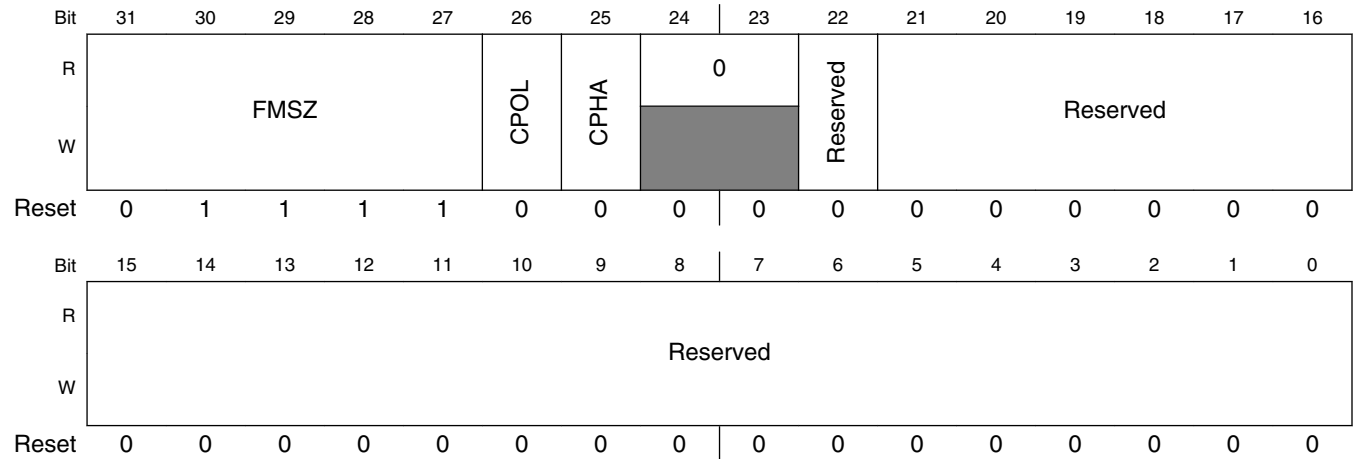
**SPIx\_CTARn field descriptions (continued)**

Field	Description
<b>Table 38-38. Baud Rate Scaler (continued)</b>	
CTARn[BR]	Baud Rate Scaler Value
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

**38.3.4 Clock and Transfer Attributes Register (In Slave Mode) (SPIx\_CTARn\_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\_CTARn\_SLAVE field descriptions**

Field	Description
31–27 FMSZ	<p>Frame Size</p> <p>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.</p>

Table continues on the next page...

## SPIx\_CTARn\_SLAVE field descriptions (continued)

Field	Description
26 CPOL	<p>Clock Polarity</p> <p>Selects the inactive state of the Serial Communications Clock (SCK).</p> <p><b>NOTE:</b> In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed.</p> <p>0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.</p>
25 CPHA	<p>Clock Phase</p> <p>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.</p> <p>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.</p>
24–23 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
22 Reserved	<p>This field is reserved.</p>
21–0 Reserved	<p>This field is reserved.</p>

### 38.3.5 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	RDFD	0
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).

Table continues on the next page...



## SPIx\_SR field descriptions (continued)

Field	Description
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  Provides a method for the module to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request.  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.
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.

Table continues on the next page...

## SPIx\_SR field descriptions (continued)

Field	Description
17 RFDF	<p>Receive FIFO Drain Flag</p> <p>Provides a method for the module to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty.</p> <p>0 RX FIFO is empty. 1 RX FIFO is not empty.</p>
16 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
15–12 TXCTR	<p>TX FIFO Counter</p> <p>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.</p>
11–8 TXNXPTR	<p>Transmit Next Pointer</p> <p>Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.</p>
7–4 RXCTR	<p>RX FIFO Counter</p> <p>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.</p>
3–0 POPXPTR	<p>Pop Next Pointer</p> <p>Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPXPTR is updated when the POPR is read.</p>

### 38.3.6 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	0	Reserved	EOQF_RE	TFUF_RE	Reserved	TFFF_RE	TFFF_DIRS	0	Reserved	Reserved	Reserved	RFOF_RE	0	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	0	0	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	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	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	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	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved.
21 Reserved	This field is reserved.
20 Reserved	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	This field is reserved. This read-only field is reserved and always has the value 0.
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  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.

*Table continues on the next page...*

**SPIx\_RSER field descriptions (continued)**

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

**38.3.7 PUSH TX FIFO Register In Master Mode (SPIx\_PUSHR)**

Specifies data to be transferred to the TX FIFO. An 8- or 16-bit write access transfers all 32 bits to the TX FIFO. In Master mode, the register transfers 16 bits of data and 16 bits of command information. In Slave mode, all 32 bits can be used as data, supporting up to 32-bit frame operation.

A read access of PUSHR returns the topmost TX FIFO entry.

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CONT		CTAS				EOQ	CTCNT	0	0	PCS					
W	CONT		CTAS				EOQ	CTCNT	0	0	PCS					
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	TXDATA															
W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**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.

*Table continues on the next page...*

**SPIx\_PUSHR field descriptions (continued)**

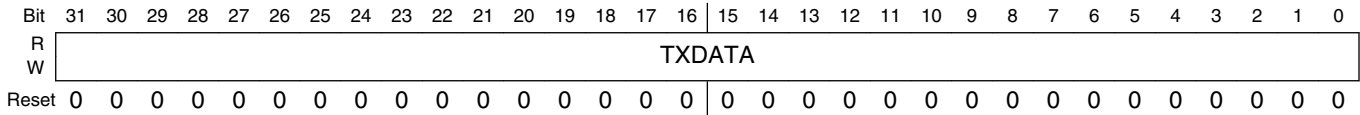
Field	Description
	0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
30–28 CTAS	Clock and Transfer Attributes Select  Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI Slave mode, CTAR0 is used. See the chip configuration 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	This field is reserved. This read-only field is reserved and always has the value 0.
23–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–16 PCS	Select which PCS signals are to be asserted for the transfer. Refer to the chip configuration details for the number of PCS signals used in this MCU.  0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
15–0 TXDATA	Transmit Data  Holds SPI data to be transferred according to the associated SPI command.

**38.3.8 PUSH TX FIFO Register In Slave Mode (SPIx\_PUSHR\_SLAVE)**

Specifies data to be transferred to the TX FIFO. An 8- or 16-bit write access to PUSHR transfers all 32 bits to the TX FIFO.

In master mode, the register transfers 16 bits of data and 16 bits of command information to the TX FIFO. In slave mode, all 32 register bits can be used as data, supporting up to 32-bit SPI Frame operation.

Address: 4002\_C000h base + 34h offset = 4002\_C034h



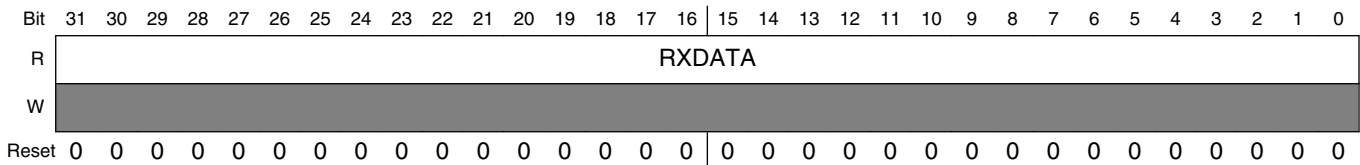
**SPIx\_PUSHR\_SLAVE field descriptions**

Field	Description
31–0 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

**38.3.9 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



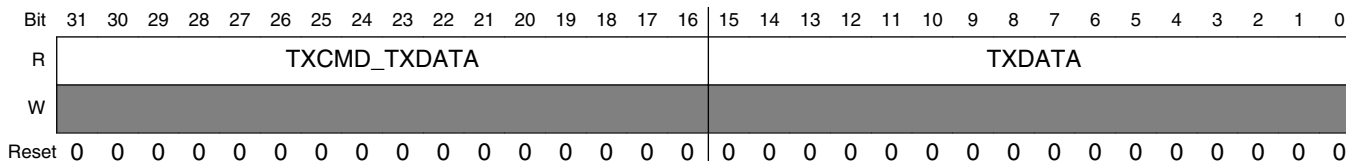
**SPIx\_POPR field descriptions**

Field	Description
31–0 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

### 38.3.10 Transmit FIFO Registers (SPIx\_TXFRn)

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



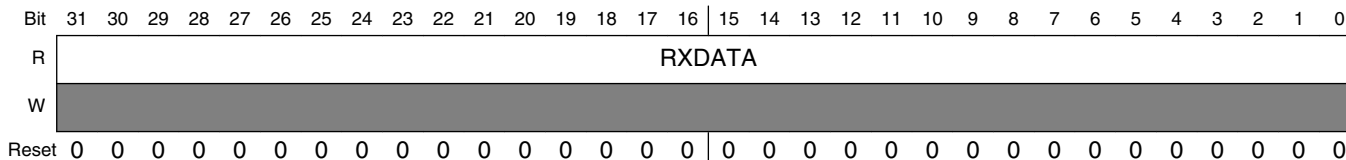
#### SPIx\_TXFRn field descriptions

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, the TXDATA contains 16 MSB bits of the SPI data to be shifted out.
15–0 TXDATA	Transmit Data  Contains the SPI data to be shifted out.

### 38.3.11 Receive FIFO Registers (SPIx\_RXFRn)

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\_RXFRn field descriptions

Field	Description
31–0 RXDATA	Receive Data  Contains the received SPI data.



## SPIx\_RXFRn field descriptions (continued)

Field	Description
-------	-------------

### 38.4 Functional description

The module supports full-duplex, synchronous serial communications between MCUs 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 configurations

- 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 \(In Master Mode\) \(SPI\\_CTARn\)](#) 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.

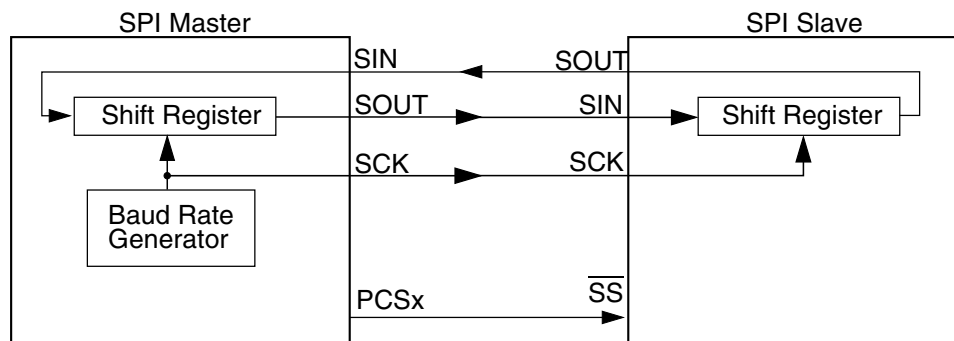


Figure 38-47. Serial protocol overview

Generally, more than one slave device can be connected to the module master. 6 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 configuration details for the number of PCS signals used in this MCU.

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](#).

### **38.4.1 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
- MCU 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
- MCU 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.

## 38.4.2 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](#)
- [Transmit First In First Out \(TX FIFO\) 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.

### 38.4.2.1 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.

### 38.4.2.2 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.

### 38.4.2.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX 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 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 PUSHHR and received data is read from the POPR.

When the TX FIFO is 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.

### 38.4.2.4 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 (PUSHHR). 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 PUSHHR[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.

#### 38.4.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO by writing to the PUSHHR. 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 PUSHHR 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.

#### 38.4.2.4.2 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.

#### 38.4.2.5 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.

### 38.4.2.5.1 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.

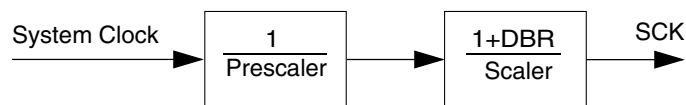
### 38.4.2.5.2 Draining the RX FIFO

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.

## 38.4.3 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol 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 38-48. Communications clock prescalers and scalers**

### 38.4.3.1 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 38-66. 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.

### 38.4.3.2 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 38-50](#) for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR<sub>x</sub> 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 38-67. PCS to SCK delay computation example**

$f_{sys}$	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 $\mu$ s

#### 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.

### 38.4.3.3 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 38-50](#) and [Figure 38-51](#) 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 38-68. After SCK Delay computation example**

$f_p$	PASC	Prescaler	ASC	Scaler	After SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 $\mu$ s

#### 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.

### 38.4.3.4 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 38-50](#) 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 38-69. Delay after Transfer computation example**

$f_p$	PDT	Prescaler	DT	Scaler	Delay after Transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

#### NOTE

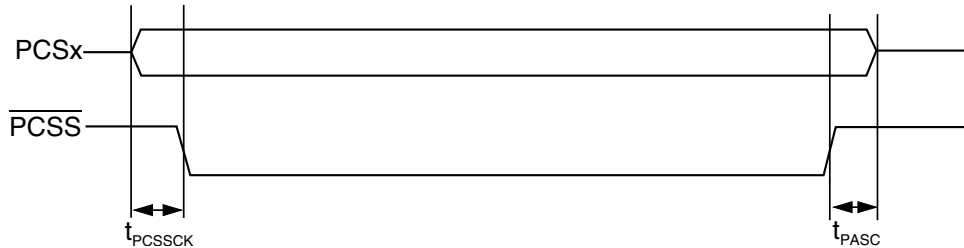
The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

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.



### 38.4.3.5 Peripheral Chip Select Strobe Enable ( $\overline{\text{PCSS}}$ )

The  $\overline{\text{PCSS}}$  signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the Module is in Master mode and the PCSSE bit is set in the MCR,  $\overline{\text{PCSS}}$  provides a signal for an external demultiplexer to decode peripheral chip selects other than PCS5 into glitch-free PCS signals. The following figure shows the timing of the  $\overline{\text{PCSS}}$  signal relative to PCS signals.



**Figure 38-49. Peripheral Chip Select Strobe timing**

The delay between the assertion of the PCS signals and the assertion of  $\overline{\text{PCSS}}$  is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{\text{PCSSCK}} = \frac{1}{f_{\text{P}}} \times \text{PCSSCK}$$

At the end of the transfer, the delay between  $\overline{\text{PCSS}}$  negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

$$t_{\text{PASC}} = \frac{1}{f_{\text{P}}} \times \text{PASC}$$

The following table shows an example of how to compute the  $t_{\text{pcssck}}$  delay.

**Table 38-70. Peripheral Chip Select Strobe Assert computation example**

$f_{\text{P}}$	PCSSCK	Prescaler	Delay before Transfer
100 MHz	0b11	7	70.0 ns

The following table shows an example of how to compute the  $t_{\text{pasc}}$  delay.

**Table 38-71. Peripheral Chip Select Strobe Negate computation example**

$f_{\text{P}}$	PASC	Prescaler	Delay after Transfer
100 MHz	0b11	7	70.0 ns

The  $\overline{\text{PCSS}}$  signal is not supported when Continuous Serial Communication SCK mode is enabled.

**NOTE**

The clock frequency mentioned in the preceding tables is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

**38.4.4 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:

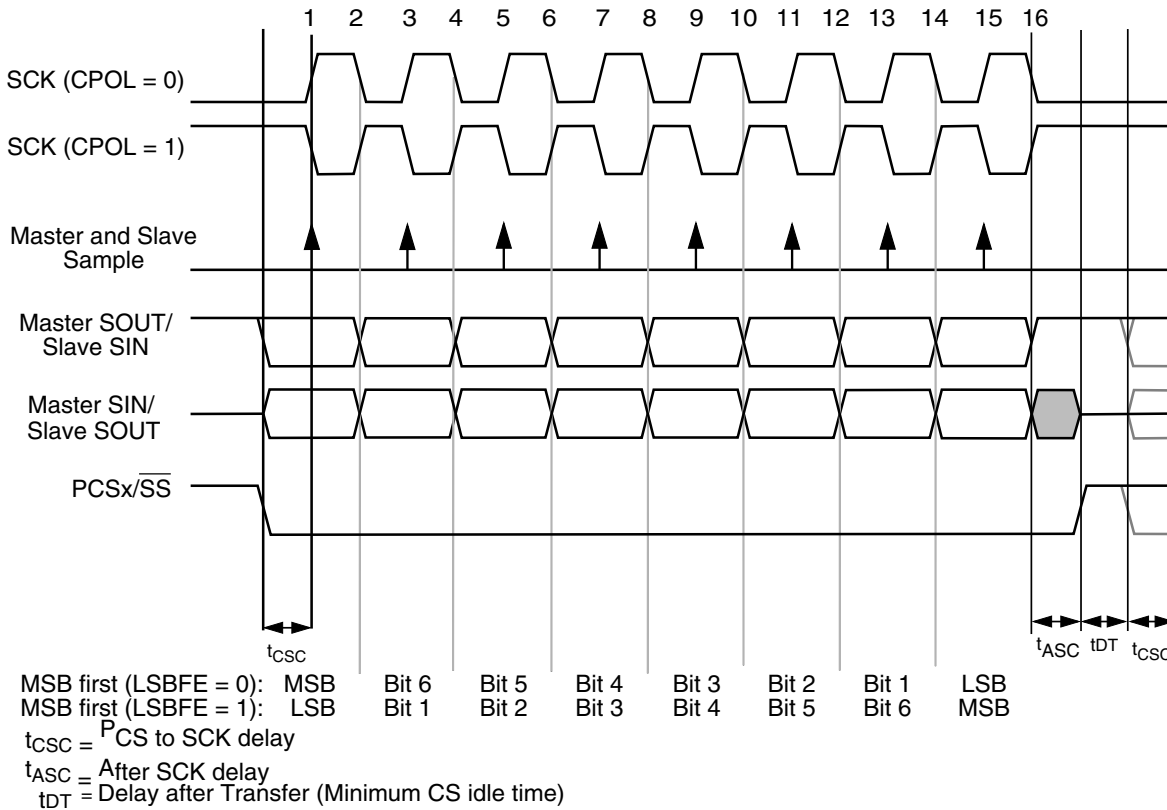
- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer Format with CPHA = 0
- Modified Transfer Format with CPHA = 1

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.

### 38.4.4.1 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.

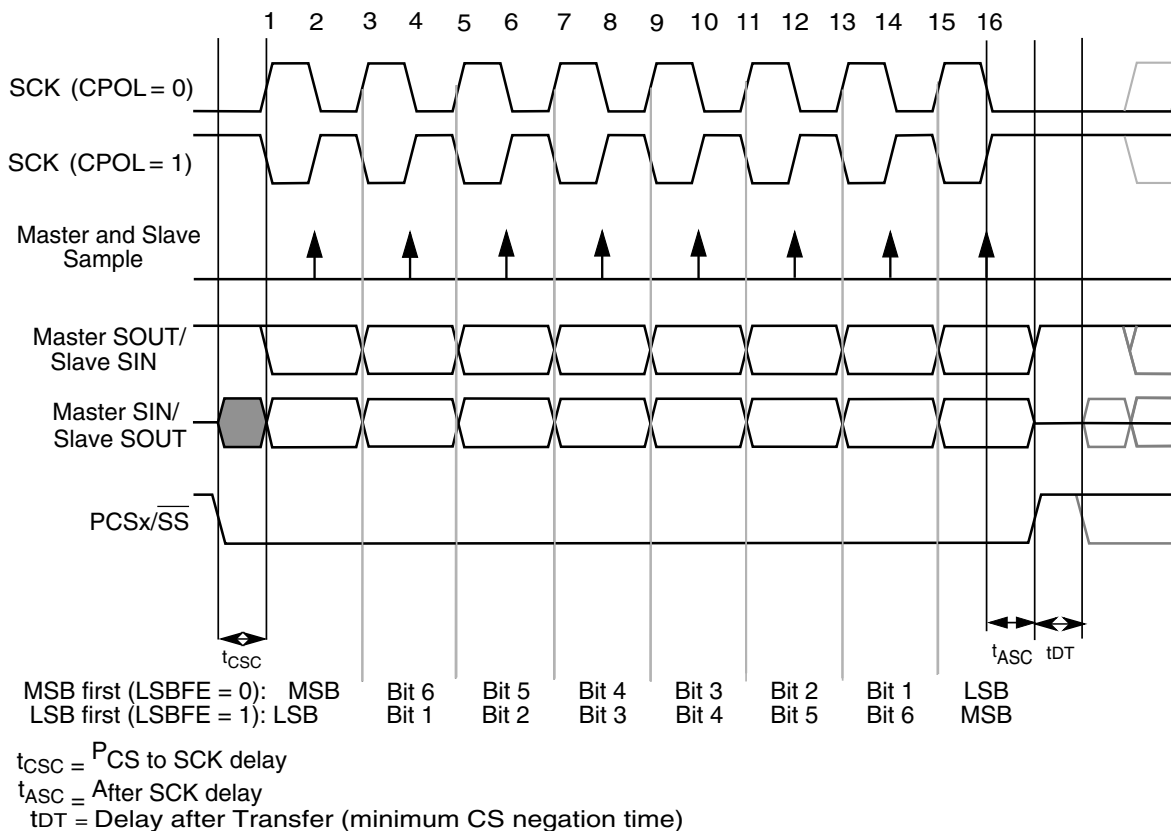


**Figure 38-50. Module transfer timing diagram (MTFE=0, CPHA=0, FMSZ=8)**

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  $t_{ASC}$  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.

### 38.4.4.2 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 38-51. Module transfer timing diagram (MTFE=0, CPHA=1, FMSZ=8)**

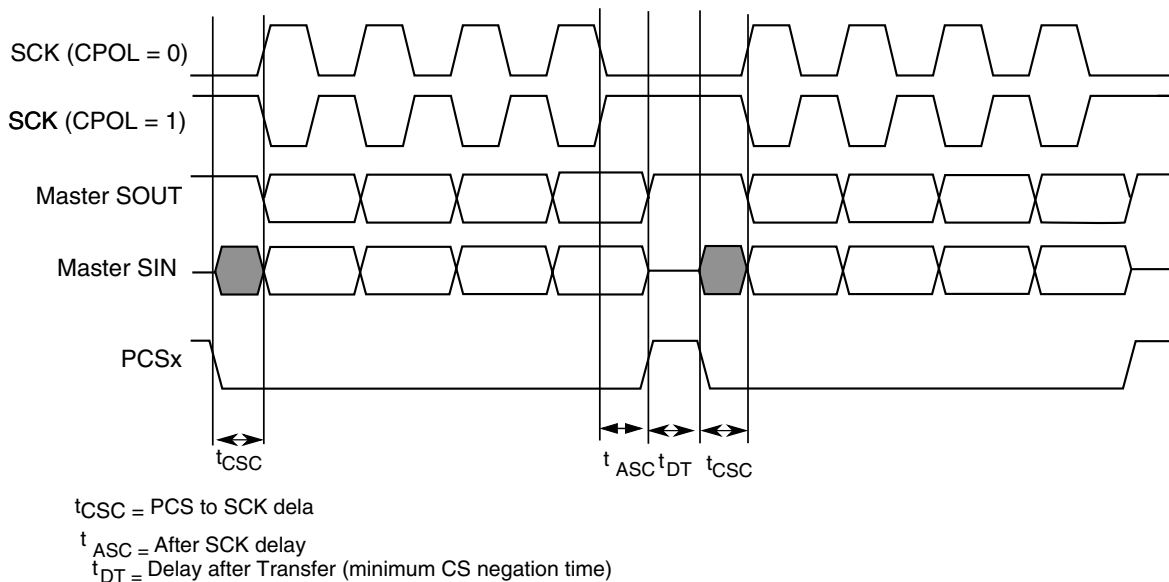
The master initiates the transfer by asserting the PCS signal to the slave. After the  $t_{CSC}$  delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave 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.

### 38.4.4.3 Continuous Selection Format

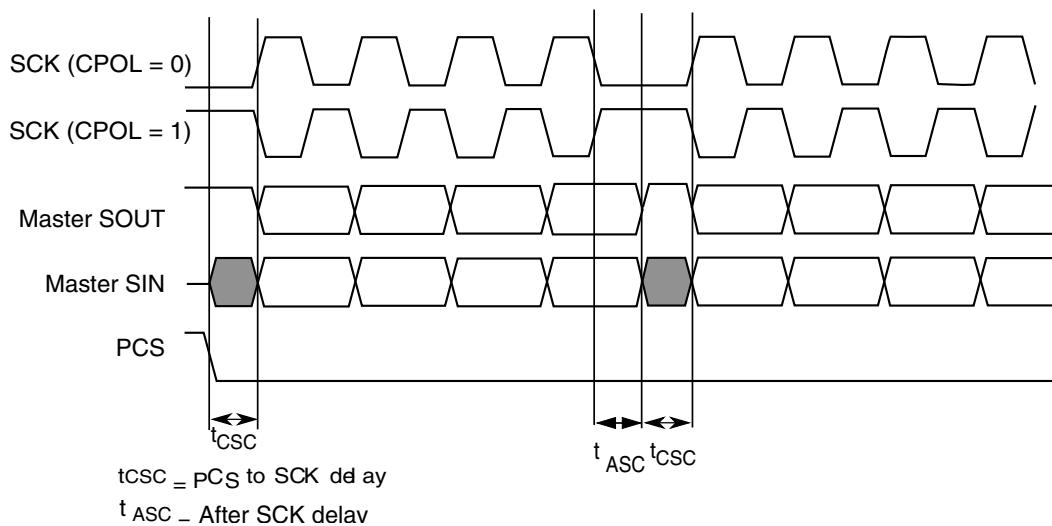
Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle the following case. The Continuous Selection Format is enabled for the SPI configuration by 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 38-52. 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 38-53. 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.

### 38.4.5 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 regardless of the MCR[HALT] bit status. 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.

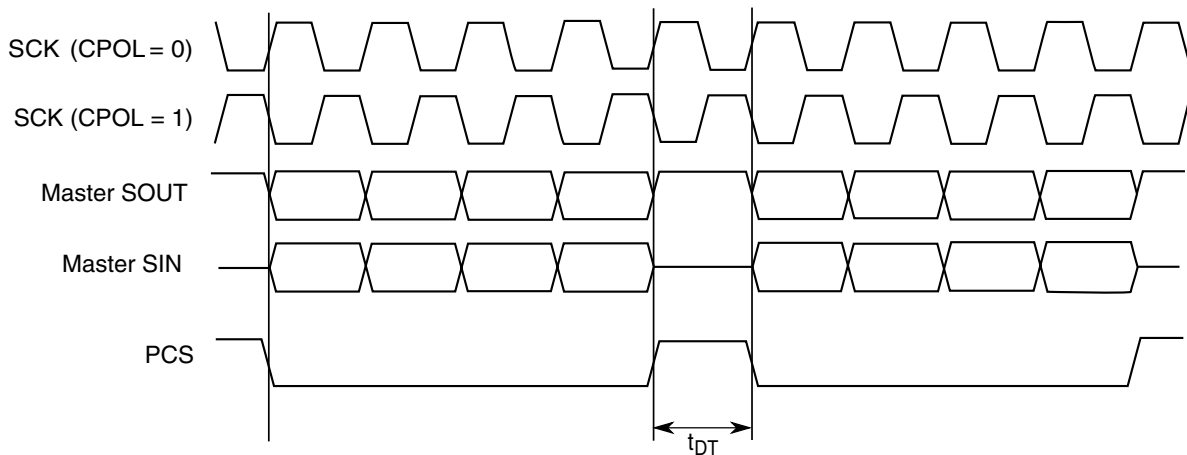
It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

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.

## Functional description

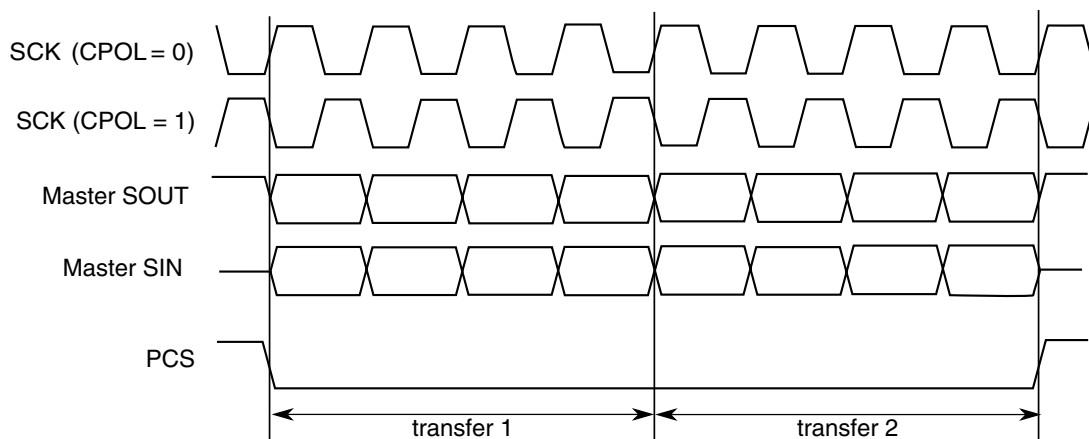


**Figure 38-54. Continuous SCK Timing Diagram (CONT=0)**

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 38-55. Continuous SCK timing diagram (CONT=1)**



### 38.4.6 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.

### 38.4.7 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 38-72. 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	-
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.

### 38.4.7.1 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.

### 38.4.7.2 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.

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.

### 38.4.7.3 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.

### 38.4.7.4 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.

### 38.4.7.5 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.

### 38.4.7.6 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.

## 38.4.8 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

### 38.4.8.1 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.

### **38.4.8.2 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. 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.

## **38.5 Initialization/application information**

This section describes how to initialize the module.

### **38.5.1 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 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 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 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.

### 38.5.2 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].

### 38.5.3 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.

### 38.5.4 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 38-73. 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	

### 38.5.5 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 38-74. 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 $\mu$ s
	32	320.0 ns	960.0 ns	1.6 $\mu$ s	2.2 $\mu$ s
	64	640.0 ns	1.9 $\mu$ s	3.2 $\mu$ s	4.5 $\mu$ s
	128	1.3 $\mu$ s	3.8 $\mu$ s	6.4 $\mu$ s	9.0 $\mu$ s
	256	2.6 $\mu$ s	7.7 $\mu$ s	12.8 $\mu$ s	17.9 $\mu$ s
	512	5.1 $\mu$ s	15.4 $\mu$ s	25.6 $\mu$ s	35.8 $\mu$ s
	1024	10.2 $\mu$ s	30.7 $\mu$ s	51.2 $\mu$ s	71.7 $\mu$ s
	2048	20.5 $\mu$ s	61.4 $\mu$ s	102.4 $\mu$ s	143.4 $\mu$ s
	4096	41.0 $\mu$ s	122.9 $\mu$ s	204.8 $\mu$ s	286.7 $\mu$ s
	8192	81.9 $\mu$ s	245.8 $\mu$ s	409.6 $\mu$ s	573.4 $\mu$ s
	16384	163.8 $\mu$ s	491.5 $\mu$ s	819.2 $\mu$ s	1.1 ms
	32768	327.7 $\mu$ s	983.0 $\mu$ s	1.6 ms	2.3 ms
65536	655.4 $\mu$ s	2.0 ms	3.3 ms	4.6 ms	

### 38.5.6 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 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](#) and [Receive First In First Out \(RX FIFO\) buffering mechanism](#) for details on the FIFO operation.

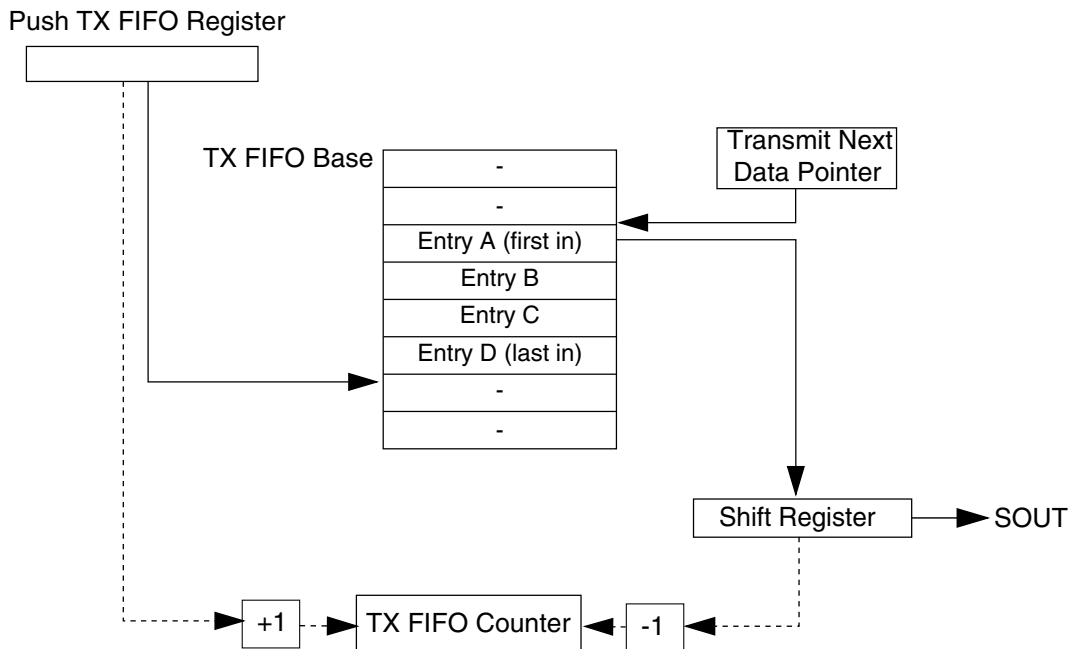


Figure 38-56. TX FIFO pointers and counter

### 38.5.6.1 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:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXPTR})$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXPTR} - 1) \bmod (\text{TXFIFOdepth})$$

TX FIFO Base - Base address of TX FIFO

TXCTR - TX FIFO Counter

TXNXPTR - Transmit Next Pointer

TX FIFO Depth - Transmit FIFO depth, implementation specific



### 38.5.6.2 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:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + (4 \times \text{POPNextPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times (\text{RXCTR} + \text{POPNextPTR} - 1) \bmod (\text{RXFIFOdepth})$$

RX FIFO Base - Base address of RX FIFO

RXCTR - RX FIFO counter

POPNextPTR - Pop Next Pointer

RX FIFO Depth - Receive FIFO depth, implementation specific



# Chapter 39

## Inter-Integrated Circuit (I2C)

### 39.1 Introduction

#### NOTE

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

The inter-integrated circuit (I<sup>2</sup>C, 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*.

#### 39.1.1 Features

The I2C module has the following features:

- Compatible with *The I<sup>2</sup>C-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

### 39.1.2 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.

### 39.1.3 Block diagram

The following figure is a functional block diagram of the I2C module.

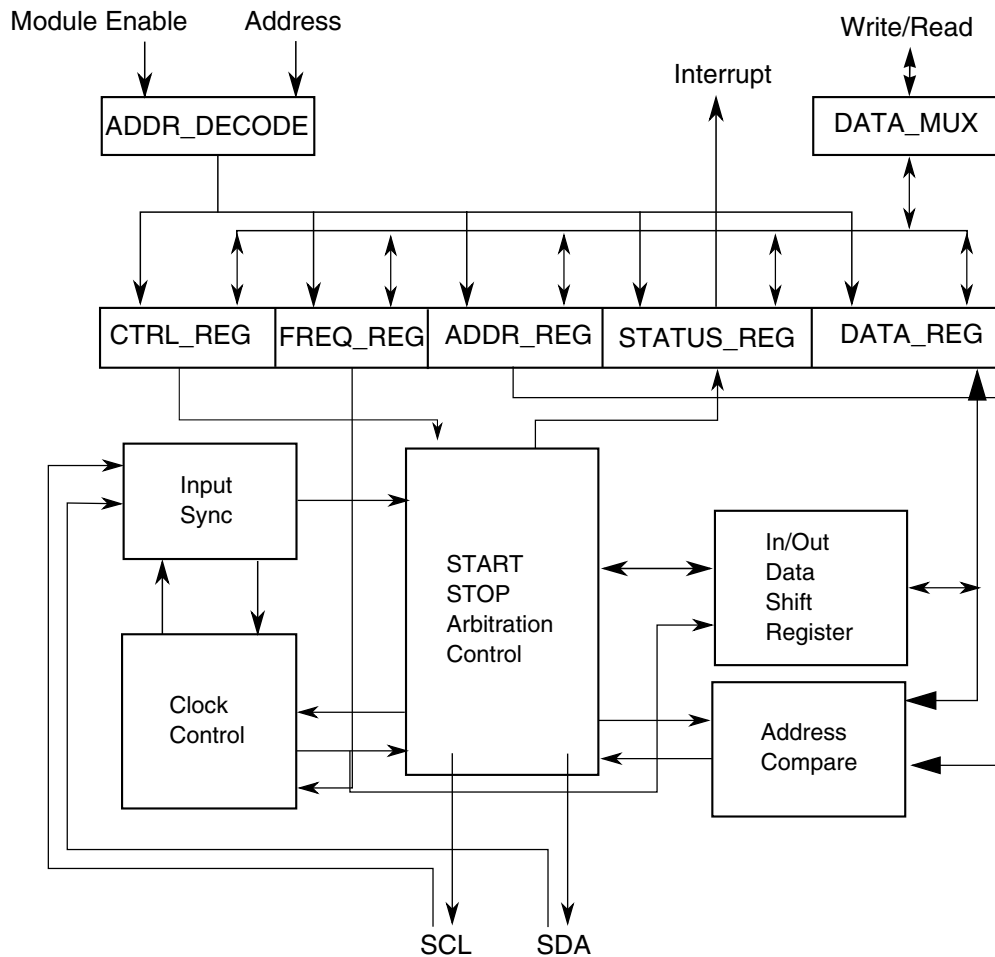


Figure 39-1. I2C Functional block diagram

## 39.2 I<sup>2</sup>C signal descriptions

The signal properties of I<sup>2</sup>C are shown in the table found here.

Table 39-1. I<sup>2</sup>C signal descriptions

Signal	Description	I/O
SCL	Bidirectional serial clock line of the I <sup>2</sup> C system.	I/O
SDA	Bidirectional serial data line of the I <sup>2</sup> C system.	I/O

## 39.3 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	<a href="#">39.3.1/926</a>
4006_6001	I2C Frequency Divider register (I2C0_F)	8	R/W	00h	<a href="#">39.3.2/927</a>
4006_6002	I2C Control Register 1 (I2C0_C1)	8	R/W	00h	<a href="#">39.3.3/928</a>
4006_6003	I2C Status register (I2C0_S)	8	R/W	80h	<a href="#">39.3.4/930</a>
4006_6004	I2C Data I/O register (I2C0_D)	8	R/W	00h	<a href="#">39.3.5/931</a>
4006_6005	I2C Control Register 2 (I2C0_C2)	8	R/W	00h	<a href="#">39.3.6/932</a>
4006_6006	I2C Programmable Input Glitch Filter Register (I2C0_FLT)	8	R/W	00h	<a href="#">39.3.7/933</a>
4006_6007	I2C Range Address register (I2C0_RA)	8	R/W	00h	<a href="#">39.3.8/934</a>
4006_6008	I2C SMBus Control and Status register (I2C0_SMB)	8	R/W	00h	<a href="#">39.3.9/935</a>
4006_6009	I2C Address Register 2 (I2C0_A2)	8	R/W	C2h	<a href="#">39.3.10/937</a>
4006_600A	I2C SCL Low Timeout Register High (I2C0_SLTH)	8	R/W	00h	<a href="#">39.3.11/937</a>
4006_600B	I2C SCL Low Timeout Register Low (I2C0_SLTL)	8	R/W	00h	<a href="#">39.3.12/937</a>

### 39.3.1 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.

Table continues on the next page...

## I2Cx\_A1 field descriptions (continued)

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

## 39.3.2 I2C Frequency Divider register (I2Cx\_F)

Address: 4006\_6000h base + 1h offset = 4006\_6001h

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

## I2Cx\_F field descriptions

Field	Description
7–6 MULT	<p>Multiplier Factor</p> <p>Defines the multiplier factor (mul). This factor is used along with the SCL divider to generate the I2C baud rate.</p> <p>00 mul = 1 01 mul = 2 10 mul = 4 11 Reserved</p>
5–0 ICR	<p>ClockRate</p> <p>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 <a href="#">I2C divider and hold values</a>.</p> <p>The SCL divider multiplied by multiplier factor (mul) determines the I2C baud rate.</p> $\text{I2C baud rate} = \text{I2C module clock speed (Hz)} / (\text{mul} \times \text{SCL divider})$ <p>The SDA hold time is the delay from the falling edge of SCL (I2C clock) to the changing of SDA (I2C data).</p> $\text{SDA hold time} = \text{I2C module clock period (s)} \times \text{mul} \times \text{SDA hold value}$ <p>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).</p> $\text{SCL start hold time} = \text{I2C module clock period (s)} \times \text{mul} \times \text{SCL start hold value}$ <p>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).</p> $\text{SCL stop hold time} = \text{I2C module clock period (s)} \times \text{mul} \times \text{SCL stop hold value}$ <p>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<sup>2</sup>C baud rate of 100 kbit/s.</p>

Table continues on the next page...

**I2Cx\_F field descriptions (continued)**

Field	Description				
	MULT	ICR	Hold times (µs)		
			SDA	SCL Start	SCL Stop
	2h	00h	3.500	3.000	5.500
	1h	07h	2.500	4.000	5.250
	1h	0Bh	2.250	4.000	5.250
	0h	14h	2.125	4.250	5.125
	0h	18h	1.125	4.750	5.125

**39.3.3 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

*Table continues on the next page...*



## I2Cx\_C1 field descriptions (continued)

Field	Description
	<p>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.</p> <p>0 Receive 1 Transmit</p>
3 TXAK	<p>Transmit Acknowledge Enable</p> <p>Specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. The value of SMB[FAACK] affects NACK/ACK generation.</p> <p><b>NOTE:</b> SCL is held low until TXAK is written.</p> <p>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).</p>
2 RSTA	<p>Repeat START</p> <p>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.</p>
1 WUEN	<p>Wakeup Enable</p> <p>The I2C module can wake the MCU from low power mode with no peripheral bus running when slave address matching occurs.</p> <p>0 Normal operation. No interrupt generated when address matching in low power mode. 1 Enables the wakeup function in low power mode.</p>
0 DMAEN	<p>DMA Enable</p> <p>Enables or disables the DMA function.</p> <p>0 All DMA signalling disabled. 1 DMA transfer is enabled. While SMB[FAACK] = 0, the following conditions trigger the DMA request:</p> <ul style="list-style-type: none"> <li>• a data byte is received, and either address or data is transmitted. (ACK/NACK is automatic)</li> <li>• the first byte received matches the A1 register or is a general call address.</li> </ul> <p>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.</p> <p>When FACK = 1, an address or a data byte is transmitted.</p>

### 39.3.4 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	<p>Transfer Complete Flag</p> <p>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.</p> <p>0 Transfer in progress 1 Transfer complete</p>
6 IAAS	<p>Addressed As A Slave</p> <p>This bit is set by one of the following conditions:</p> <ul style="list-style-type: none"> <li>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).</li> <li>C2[GCAEN] is set and a general call is received.</li> <li>SMB[SIICAEN] is set and the calling address matches the second programmed slave address.</li> <li>ALERTEN is set and an SMBus alert response address is received</li> <li>RMEN is set and an address is received that is within the range between the values of the A1 and RA registers.</li> </ul> <p>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.</p> <p>0 Not addressed 1 Addressed as a slave</p>
5 BUSY	<p>Bus Busy</p> <p>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.</p> <p>0 Bus is idle 1 Bus is busy</p>
4 ARBL	<p>Arbitration Lost</p> <p>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.</p> <p>0 Standard bus operation. 1 Loss of arbitration.</p>
3 RAM	<p>Range Address Match</p>

Table continues on the next page...

## I2Cx\_S field descriptions (continued)

Field	Description
	<p>This bit is set to 1 by any of the following conditions, if I2C_C2[RMEN] = 1:</p> <ul style="list-style-type: none"> <li>Any nonzero calling address is received that matches the address in the RA register.</li> <li>The calling address is within the range of values of the A1 and RA registers.</li> </ul> <p><b>NOTE:</b> For the RAM bit to be set to 1 correctly, C1[IICIE] must be set to 1.</p> <p>Writing the C1 register with any value clears this bit to 0.</p> <p>0 Not addressed 1 Addressed as a slave</p>
2 SRW	<p>Slave Read/Write</p> <p>When addressed as a slave, SRW indicates the value of the R/W command bit of the calling address sent to the master.</p> <p>0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave</p>
1 IICIF	<p>Interrupt Flag</p> <p>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:</p> <ul style="list-style-type: none"> <li>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.</li> <li>One byte transfer, excluding ACK/NACK bit, completes if FACK is 1.</li> <li>Match of slave address to calling address including primary slave address, range slave address, alert response address, second slave address, or general call address.</li> <li>Arbitration lost</li> <li>In SMBus mode, any timeouts except SCL and SDA high timeouts</li> <li>I2C bus stop or start detection if the SSIE bit in the Input Glitch Filter register is 1</li> </ul> <p><b>NOTE:</b> 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.</p> <p>0 No interrupt pending 1 Interrupt pending</p>
0 RXAK	<p>Receive Acknowledge</p> <p>0 Acknowledge signal was received after the completion of one byte of data transmission on the bus 1 No acknowledge signal detected</p>

## 39.3.5 I2C Data I/O register (I2Cx\_D)

Address: 4006\_6000h base + 4h offset = 4006\_6004h

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

### I2Cx\_D field descriptions

Field	Description
7-0 DATA	<p>Data</p> <p>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.</p> <p><b>NOTE:</b> 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.</p> <p>In slave mode, the same functions are available after an address match occurs.</p> <p>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.</p> <p>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.</p> <p>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/W bit (in position bit 0).</p>

### 39.3.6 I2C Control Register 2 (I2Cx\_C2)

Address: 4006\_6000h base + 5h offset = 4006\_6005h

Bit	7	6	5	4	3	2	1	0
Read	GCAEN	ADEXT	HDRS	SBRC	RMEN	AD[10:8]		
Write								
Reset	0	0	0	0	0	0	0	0

### I2Cx\_C2 field descriptions

Field	Description
7 GCAEN	<p>General Call Address Enable</p> <p>Enables general call address.</p> <p>0 Disabled 1 Enabled</p>
6 ADEXT	<p>Address Extension</p> <p>Controls the number of bits used for the slave address.</p> <p>0 7-bit address scheme 1 10-bit address scheme</p>
5 HDRS	<p>High Drive Select</p> <p>Controls the drive capability of the I2C pads.</p>

Table continues on the next page...

**I2Cx\_C2 field descriptions (continued)**

Field	Description
	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
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.
2-0 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.

**39.3.7 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: <ol style="list-style-type: none"> <li>1. The I2C module is configured for a basic transfer, and the SHEN bit is set to 1.</li> <li>2. A transfer begins.</li> <li>3. The MCU signals the I2C module to enter stop mode.</li> <li>4. The byte currently being transferred, including both address and data, completes its transfer.</li> <li>5. The I2C slave or master acknowledges that the in-transfer byte completed its transfer and acknowledges the request to enter stop mode.</li> <li>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.</li> </ol>

*Table continues on the next page...*

I2Cx\_FLT field descriptions (continued)

Field	Description
	<p>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.</p> <p>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.</p> <p>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.</p> <p>0 Stop holdoff is disabled. The MCU's entry to stop mode is not gated.                      1 Stop holdoff is enabled.</p>
6 STOPF	<p>I2C Bus Stop Detect Flag</p> <p>Hardware sets this bit when the I2C bus's stop status is detected. The STOPF bit must be cleared by writing 1 to it.</p> <p>0 No stop happens on I2C bus                      1 Stop detected on I2C bus</p>
5 SSIE	<p>I2C Bus Stop or Start Interrupt Enable</p> <p>This bit enables the interrupt for I2C bus stop or start detection.</p> <p><b>NOTE:</b> 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.</p> <p>0 Stop or start detection interrupt is disabled                      1 Stop or start detection interrupt is enabled</p>
4 STARTF	<p>I2C Bus Start Detect Flag</p> <p>Hardware sets this bit when the I2C bus's start status is detected. The STARTF bit must be cleared by writing 1 to it.</p> <p>0 No start happens on I2C bus                      1 Start detected on I2C bus</p>
3-0 FLT	<p>I2C Programmable Filter Factor</p> <p>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.</p> <p>0h No filter/bypass                      1-Fh Filter glitches up to width of <i>n</i> I2C module clock cycles, where <i>n</i>=1-15d</p>

39.3.8 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  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.

## 39.3.9 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	FAACK	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 FAACK	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.

Table continues on the next page...

## I2Cx\_SMB field descriptions (continued)

Field	Description
	<p><b>NOTE:</b> 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.</p> <p>0 SMBus alert response address matching is disabled 1 SMBus alert response address matching is enabled</p>
5 SIICAEN	<p>Second I2C Address Enable</p> <p>Enables or disables SMBus device default address.</p> <p>0 I2C address register 2 matching is disabled 1 I2C address register 2 matching is enabled</p>
4 TCKSEL	<p>Timeout Counter Clock Select</p> <p>Selects the clock source of the timeout counter.</p> <p>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</p>
3 SLTF	<p>SCL Low Timeout Flag</p> <p>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.</p> <p><b>NOTE:</b> The low timeout function is disabled when the SLT register's value is 0.</p> <p>0 No low timeout occurs 1 Low timeout occurs</p>
2 SHTF1	<p>SCL High Timeout Flag 1</p> <p>This read-only bit sets when SCL and SDA are held high more than <math>\text{clock} \times \text{LoValue} / 512</math>, which indicates the bus is free. This bit is cleared automatically.</p> <p>0 No SCL high and SDA high timeout occurs 1 SCL high and SDA high timeout occurs</p>
1 SHTF2	<p>SCL High Timeout Flag 2</p> <p>This bit sets when SCL is held high and SDA is held low more than <math>\text{clock} \times \text{LoValue} / 512</math>. Software clears this bit by writing 1 to it.</p> <p>0 No SCL high and SDA low timeout occurs 1 SCL high and SDA low timeout occurs</p>
0 SHTF2IE	<p>SHTF2 Interrupt Enable</p> <p>Enables SCL high and SDA low timeout interrupt.</p> <p>0 SHTF2 interrupt is disabled 1 SHTF2 interrupt is enabled</p>



### 39.3.10 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	SAD							0
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.

### 39.3.11 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	SSLT[15:8]							
Reset	0	0	0	0	0	0	0	0

#### I2Cx\_SLTH field descriptions

Field	Description
7–0 SSLT[15:8]	SSLT[15:8] Most significant byte of SCL low timeout value that determines the timeout period of SCL low.

### 39.3.12 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	SSLT[7:0]							
Reset	0	0	0	0	0	0	0	0

**I2Cx\_SLTL field descriptions**

Field	Description
7-0 SSLT[7:0]	SSLT[7:0] Least significant byte of SCL low timeout value that determines the timeout period of SCL low.

## 39.4 Functional description

This section provides a comprehensive functional description of the I2C module.

### 39.4.1 I2C protocol

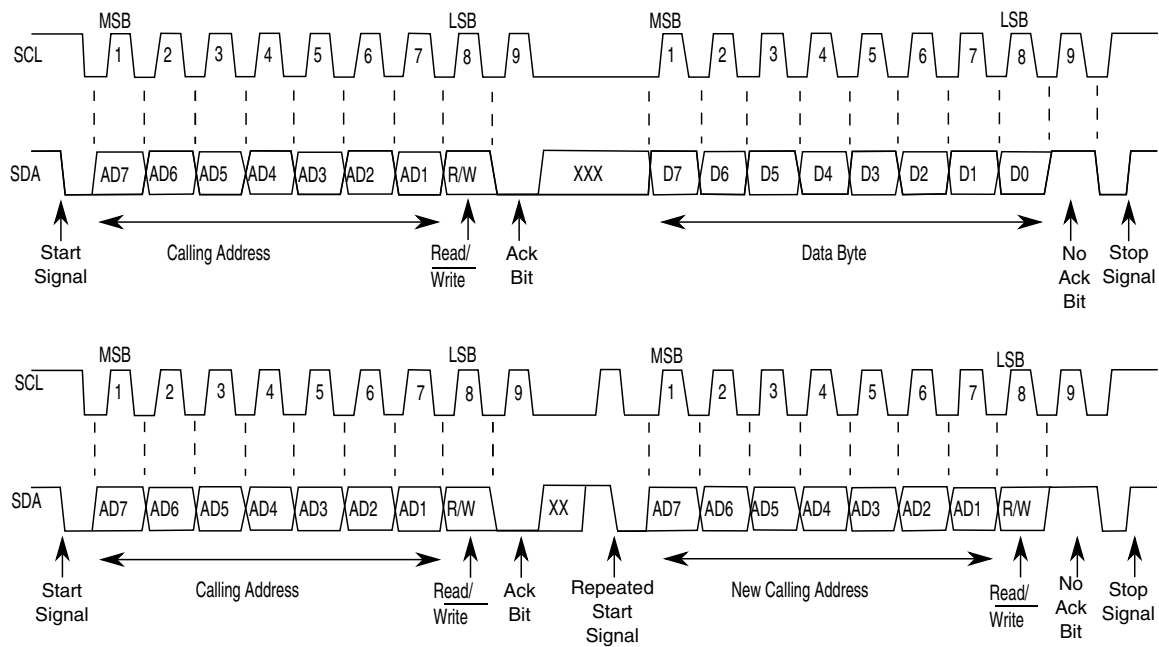
The I2C bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfers.

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 39-26. I2C bus transmission signals**

### 39.4.1.1 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.

### 39.4.1.2 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  $R/\bar{W}$  bit. The  $R/\bar{W}$  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.

### **39.4.1.3 Data transfers**

When successful slave addressing is achieved, data transfer can proceed on a byte-by-byte basis in the direction specified by the  $\overline{R/W}$  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.

### **39.4.1.4 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.

### 39.4.1.5 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.

### 39.4.1.6 Arbitration procedure

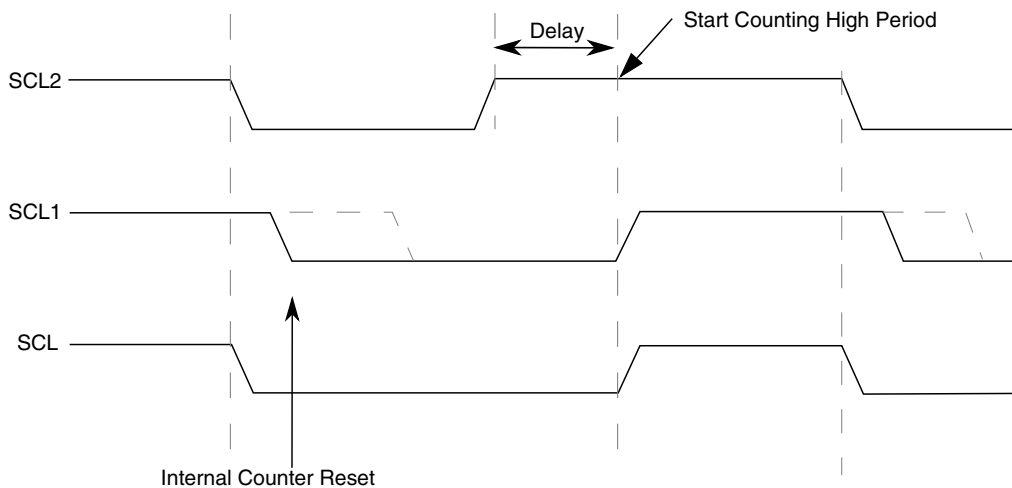
The I2C bus is a true multimaster bus that allows more than one master to be connected on it.

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.

### 39.4.1.7 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 39-27. I2C clock synchronization**

### 39.4.1.8 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.

### 39.4.1.9 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.

### 39.4.1.10 I2C divider and hold values

#### NOTE

For some cases on some devices, the SCL divider value may vary by  $\pm 2$  or  $\pm 4$  when ICR's value ranges from 00h to 0Fh. These potentially varying SCL divider values are highlighted in the following table. For the actual SCL divider values for your device, see the chip-specific details about the I2C module.

Table 39-28. 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

## 39.4.2 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.

### 39.4.2.1 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 39-29. 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.

### 39.4.2.2 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 39-30. 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.

### 39.4.3 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.

### 39.4.4 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.

#### 39.4.4.1 Timeouts

The  $T_{\text{TIMEOUT,MIN}}$  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  $T_{\text{TIMEOUT,MIN}}$ . Devices that have detected this condition must reset their communication and be able to receive a new START condition within the timeframe of  $T_{\text{TIMEOUT,MAX}}$ .

SMBus defines a clock low timeout,  $T_{\text{TIMEOUT}}$ , of 35 ms, specifies  $T_{\text{LOW:SEXT}}$  as the cumulative clock low extend time for a slave device, and specifies  $T_{\text{LOW:MEXT}}$  as the cumulative clock low extend time for a master device.

##### 39.4.4.1.1 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  $T_{\text{TIMEOUT,MIN}}$ , 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  $T_{\text{TIMEOUT,MIN}}$  condition, it resets its communication and is then able to receive a new START condition.

### 39.4.4.1.2 SCL high timeout

When the I2C module has determined that the SMBCLK and SMBDAT signals have been high for at least  $T_{\text{HIGH:MAX}}$ , 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.

### 39.4.4.1.3 CSMBCLK TIMEOUT MEXT and CSMBCLK TIMEOUT SEXT

The following figure illustrates the definition of the timeout intervals  $T_{\text{LOW:SEXT}}$  and  $T_{\text{LOW:MEXT}}$ . When in master mode, the I2C module must not cumulatively extend its clock cycles for a period greater than  $T_{\text{LOW:MEXT}}$  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.

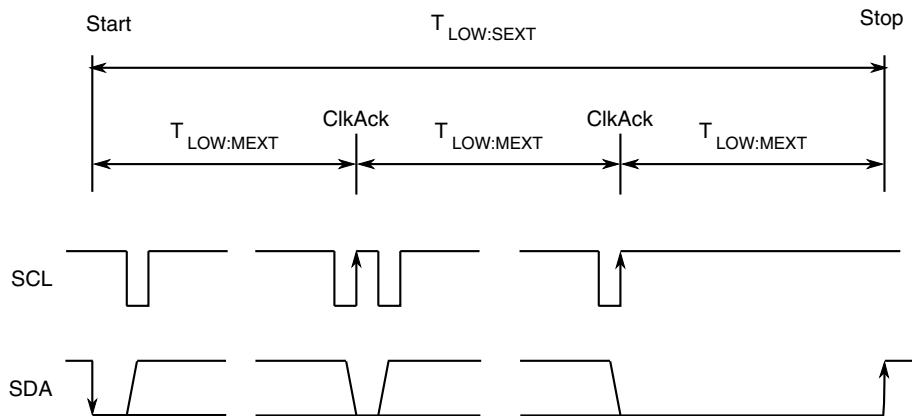


Figure 39-28. Timeout measurement intervals

A master is allowed to abort the transaction in progress to any slave that violates the  $T_{\text{LOW:SEXT}}$  or  $T_{\text{TIMEOUT,MIN}}$  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  $T_{\text{LOW:SEXT}}$  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.

**39.4.4.2 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.

**39.4.5 Resets**

The I2C module is disabled after a reset. The I2C module cannot cause a core reset.

## 39.4.6 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 39-31. 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 <sup>2</sup> C bus stop detection	STOPF	IICIF	IICIE & SSIE
I <sup>2</sup> C 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

### 39.4.6.1 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.

### 39.4.6.2 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.

### 39.4.6.3 Stop Detect Interrupt

When the stop status is detected on the I<sup>2</sup>C bus, the STOPF bit is set to 1. The CPU is interrupted, provided the IICIE and STOPIE bits are both set to 1.

### 39.4.6.4 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.

### 39.4.6.5 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.

### 39.4.6.6 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.

### 39.4.7 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.

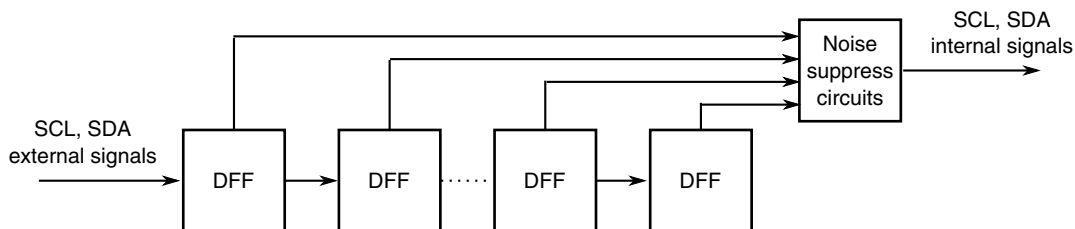


Figure 39-29. Programmable input glitch filter diagram

### 39.4.8 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.

After the address matching IAAS bit is set, an interrupt is sent at the end of address matching to wake the core.

**NOTE**

In Stop mode, the I2C module supports slave receive mode only. To avoid I2C bus conflicts during wakeup from Stop mode, software must ensure the following before entering Stop mode:

- I2C data transfers have completed.
- The I2C module is in slave receive mode (C1[MST]=0, C1[TX]=0).

**NOTE**

During the wake-up process, if an external master continues to send data to the slave, the baud rate under Stop mode must be less than 50 kbit/s. To avoid the slower baud rate under Stop mode, the master can add a short delay in firmware to wait until the wake-up process is complete and then send data.

**NOTE**

Wake-up caused by an address match is not supported for SMBus mode.

**39.4.9 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.



## 39.5 Initialization/application information

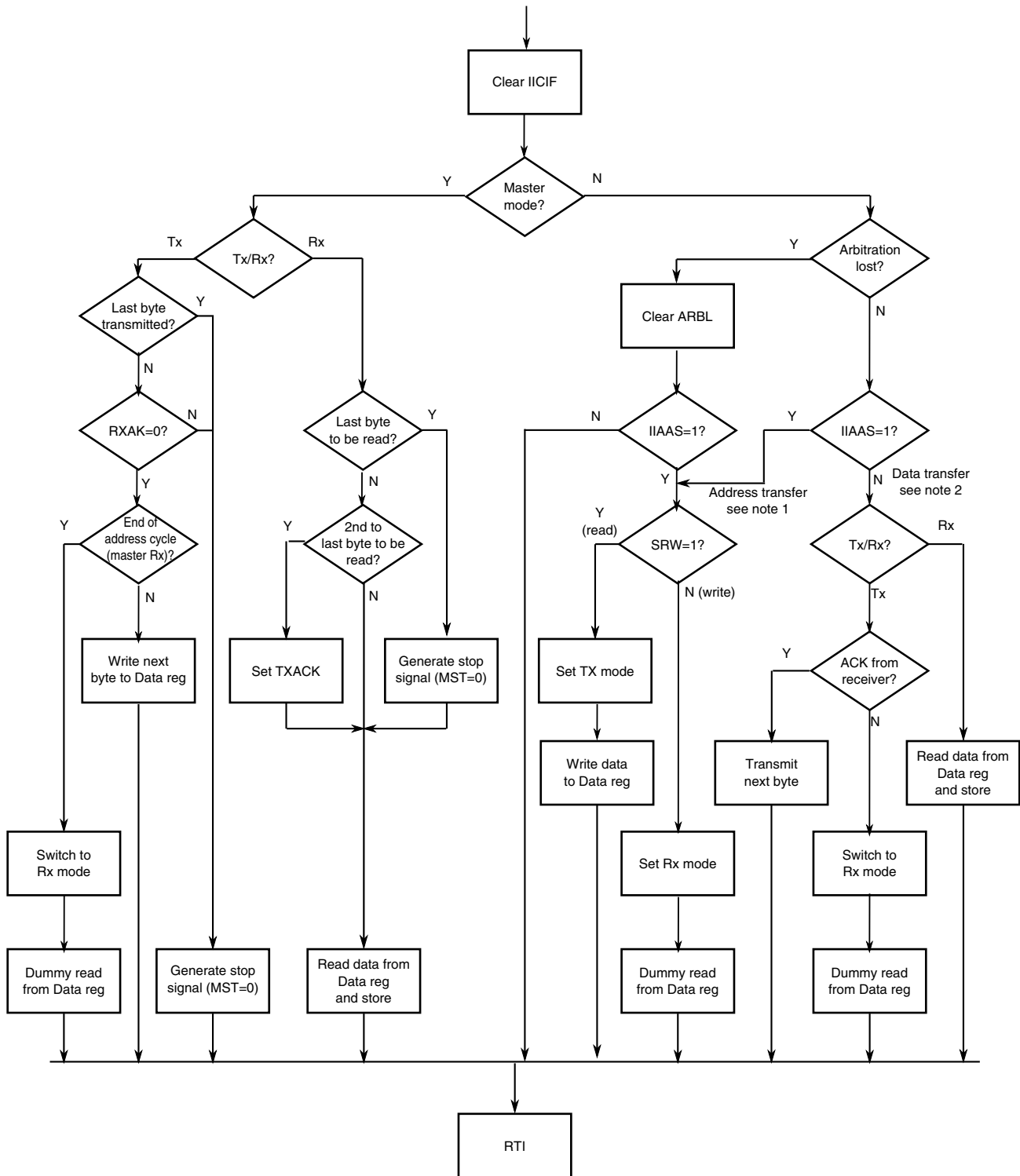
### Module Initialization (Slave)

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)

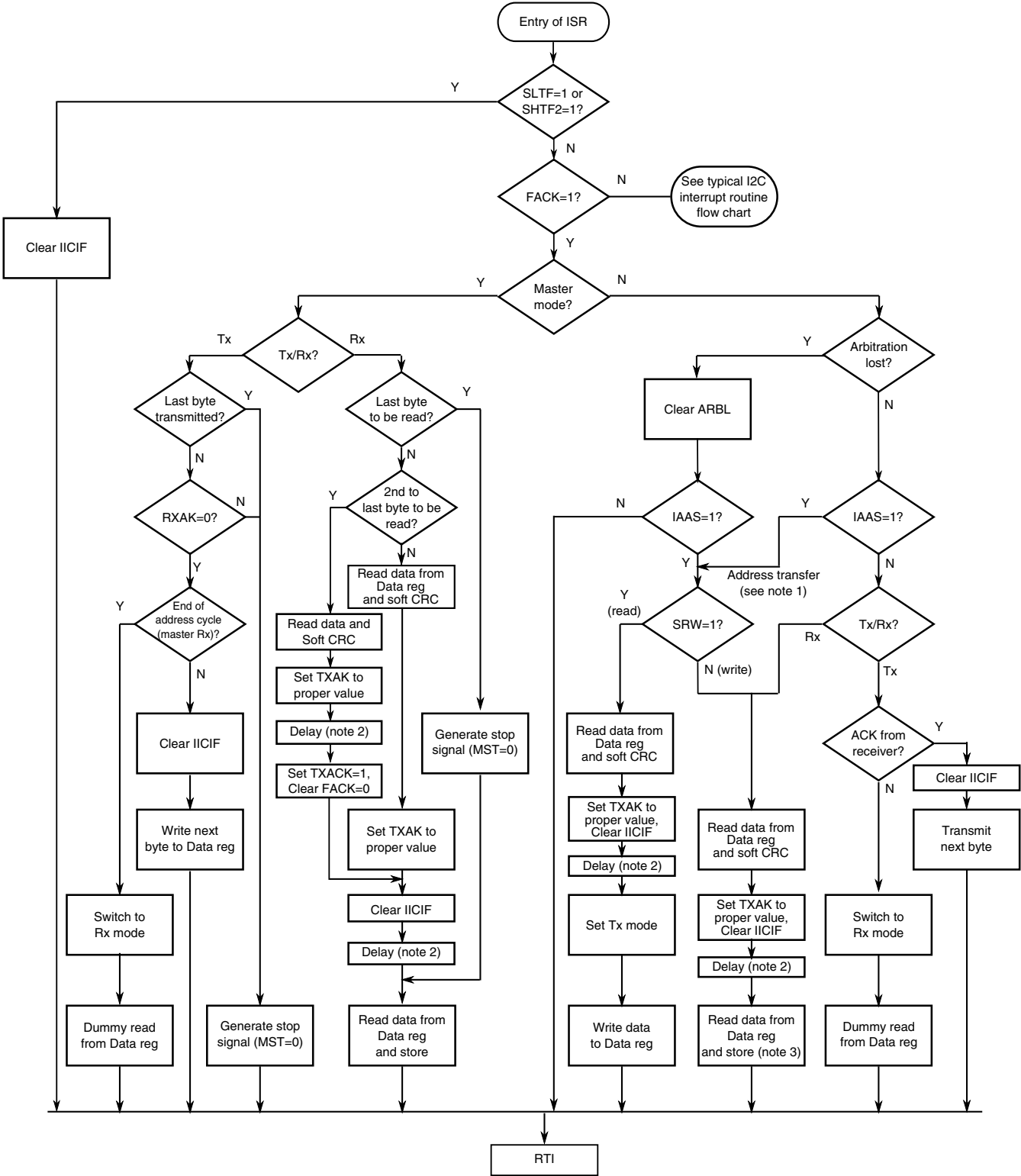
The routine shown in the following figure encompasses both master and slave I2C operations. For slave operation, an incoming I2C message that contains the proper address begins I2C communication. For master operation, communication must be initiated by writing the Data register. An example of an I2C 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 39-30. Typical I2C interrupt routine**



- Notes:**
1. If general call or SIICAE 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 39-31. Typical I2C SMBus interrupt routine



# Chapter 40

## Universal Asynchronous Receiver/Transmitter (UART)

### 40.1 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.

#### 40.1.1 Features

The UART includes the following features:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse width
- 13-bit baud rate selection with /32 fractional divide, based on the module clock frequency
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver
- Programmable transmitter output polarity
- Programmable receive input polarity
- Up to 14-bit break character transmission.

- 11-bit break character detection option
- 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
  - 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

### 40.1.2 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

### 40.1.2.1 Run mode

This is the normal mode of operation.

### 40.1.2.2 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.

### 40.1.2.3 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.

## 40.2 UART signal descriptions

The UART signals are shown in the following table.

**Table 40-1. UART signal descriptions**

Signal	Description	I/O
CTS	Clear to send	I
RTS	Request to send	O
RXD	Receive data	I
TXD	Transmit data	O

### 40.2.1 Detailed signal descriptions

The detailed signal descriptions of the UART are shown in the following table.

**Table 40-2. UART—Detailed signal descriptions**

Signal	I/O	Description				
CTS	I	Clear to send. Indicates whether the UART can start transmitting data when flow control is enabled.				
		<table border="1"> <tr> <td><b>State meaning</b></td> <td>Asserted—Data transmission can start. Negated—Data transmission cannot start.</td> </tr> <tr> <td><b>Timing</b></td> <td>Assertion—When transmitting device's RTS asserts. Negation—When transmitting device's RTS deasserts.</td> </tr> </table>	<b>State meaning</b>	Asserted—Data transmission can start. Negated—Data transmission cannot start.	<b>Timing</b>	Assertion—When transmitting device's RTS asserts. Negation—When transmitting device's RTS deasserts.
		<b>State meaning</b>	Asserted—Data transmission can start. Negated—Data transmission cannot start.			
<b>Timing</b>	Assertion—When transmitting device's RTS asserts. Negation—When transmitting device's RTS deasserts.					
<table border="1"> <tr> <td><b>State meaning</b></td> <td>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.</td> </tr> <tr> <td><b>Timing</b></td> <td>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.</td> </tr> </table>	<b>State meaning</b>	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.	<b>Timing</b>	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.		
<b>State meaning</b>	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.					
<b>Timing</b>	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.					
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.				
		<table border="1"> <tr> <td><b>State meaning</b></td> <td>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.</td> </tr> <tr> <td><b>Timing</b></td> <td>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.</td> </tr> </table>	<b>State meaning</b>	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.	<b>Timing</b>	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.
		<b>State meaning</b>	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.			
<b>Timing</b>	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.					
<table border="1"> <tr> <td><b>State meaning</b></td> <td>Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.</td> </tr> <tr> <td><b>Timing</b></td> <td>Sampled at a frequency determined by the module clock divided by the baud rate.</td> </tr> </table>	<b>State meaning</b>	Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.	<b>Timing</b>	Sampled at a frequency determined by the module clock divided by the baud rate.		
<b>State meaning</b>	Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.					
<b>Timing</b>	Sampled at a frequency determined by the module clock divided by the baud rate.					
RXD	I	Receive data. Serial data input to receiver.				
		<table border="1"> <tr> <td><b>State meaning</b></td> <td>Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.</td> </tr> <tr> <td><b>Timing</b></td> <td>Sampled at a frequency determined by the module clock divided by the baud rate.</td> </tr> </table>	<b>State meaning</b>	Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.	<b>Timing</b>	Sampled at a frequency determined by the module clock divided by the baud rate.
		<b>State meaning</b>	Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.			
<b>Timing</b>	Sampled at a frequency determined by the module clock divided by the baud rate.					
<table border="1"> <tr> <td><b>State meaning</b></td> <td>Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.</td> </tr> <tr> <td><b>Timing</b></td> <td>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.</td> </tr> </table>	<b>State meaning</b>	Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.	<b>Timing</b>	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.		
<b>State meaning</b>	Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.					
<b>Timing</b>	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.					
TXD	O	Transmit data. Serial data output from transmitter.				
		<table border="1"> <tr> <td><b>State meaning</b></td> <td>Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.</td> </tr> <tr> <td><b>Timing</b></td> <td>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.</td> </tr> </table>	<b>State meaning</b>	Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.	<b>Timing</b>	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.
		<b>State meaning</b>	Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.			
<b>Timing</b>	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.					
<table border="1"> <tr> <td><b>State meaning</b></td> <td>Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.</td> </tr> <tr> <td><b>Timing</b></td> <td>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.</td> </tr> </table>	<b>State meaning</b>	Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.	<b>Timing</b>	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.		
<b>State meaning</b>	Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.					
<b>Timing</b>	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.					



## 40.3 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	<a href="#">40.3.1/963</a>
4006_A001	UART Baud Rate Registers: Low (UART0_BDL)	8	R/W	04h	<a href="#">40.3.2/964</a>
4006_A002	UART Control Register 1 (UART0_C1)	8	R/W	00h	<a href="#">40.3.3/964</a>
4006_A003	UART Control Register 2 (UART0_C2)	8	R/W	00h	<a href="#">40.3.4/966</a>
4006_A004	UART Status Register 1 (UART0_S1)	8	R	C0h	<a href="#">40.3.5/967</a>
4006_A005	UART Status Register 2 (UART0_S2)	8	R/W	00h	<a href="#">40.3.6/970</a>
4006_A006	UART Control Register 3 (UART0_C3)	8	R/W	00h	<a href="#">40.3.7/972</a>
4006_A007	UART Data Register (UART0_D)	8	R/W	00h	<a href="#">40.3.8/973</a>
4006_A008	UART Match Address Registers 1 (UART0_MA1)	8	R/W	00h	<a href="#">40.3.9/975</a>
4006_A009	UART Match Address Registers 2 (UART0_MA2)	8	R/W	00h	<a href="#">40.3.10/975</a>
4006_A00A	UART Control Register 4 (UART0_C4)	8	R/W	00h	<a href="#">40.3.11/975</a>
4006_A00B	UART Control Register 5 (UART0_C5)	8	R/W	00h	<a href="#">40.3.12/976</a>
4006_A00C	UART Extended Data Register (UART0_ED)	8	R	00h	<a href="#">40.3.13/977</a>
4006_A00D	UART Modem Register (UART0_MODEM)	8	R/W	00h	<a href="#">40.3.14/978</a>
4006_A00E	UART Infrared Register (UART0_IR)	8	R/W	00h	<a href="#">40.3.15/979</a>
4006_A010	UART FIFO Parameters (UART0_PFIFO)	8	R/W	<a href="#">See section</a>	<a href="#">40.3.16/980</a>
4006_A011	UART FIFO Control Register (UART0_CFIFO)	8	R/W	00h	<a href="#">40.3.17/981</a>
4006_A012	UART FIFO Status Register (UART0_SFIFO)	8	R/W	C0h	<a href="#">40.3.18/982</a>
4006_A013	UART FIFO Transmit Watermark (UART0_TWFIFO)	8	R/W	00h	<a href="#">40.3.19/983</a>
4006_A014	UART FIFO Transmit Count (UART0_TCFIFO)	8	R	00h	<a href="#">40.3.20/984</a>

*Table continues on the next page...*

## UART memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4006_A015	UART FIFO Receive Watermark (UART0_RWFIFO)	8	R/W	01h	<a href="#">40.3.21/984</a>
4006_A016	UART FIFO Receive Count (UART0_RCFIFO)	8	R	00h	<a href="#">40.3.22/985</a>
4006_B000	UART Baud Rate Registers: High (UART1_BDH)	8	R/W	00h	<a href="#">40.3.1/963</a>
4006_B001	UART Baud Rate Registers: Low (UART1_BDL)	8	R/W	04h	<a href="#">40.3.2/964</a>
4006_B002	UART Control Register 1 (UART1_C1)	8	R/W	00h	<a href="#">40.3.3/964</a>
4006_B003	UART Control Register 2 (UART1_C2)	8	R/W	00h	<a href="#">40.3.4/966</a>
4006_B004	UART Status Register 1 (UART1_S1)	8	R	C0h	<a href="#">40.3.5/967</a>
4006_B005	UART Status Register 2 (UART1_S2)	8	R/W	00h	<a href="#">40.3.6/970</a>
4006_B006	UART Control Register 3 (UART1_C3)	8	R/W	00h	<a href="#">40.3.7/972</a>
4006_B007	UART Data Register (UART1_D)	8	R/W	00h	<a href="#">40.3.8/973</a>
4006_B008	UART Match Address Registers 1 (UART1_MA1)	8	R/W	00h	<a href="#">40.3.9/975</a>
4006_B009	UART Match Address Registers 2 (UART1_MA2)	8	R/W	00h	<a href="#">40.3.10/975</a>
4006_B00A	UART Control Register 4 (UART1_C4)	8	R/W	00h	<a href="#">40.3.11/975</a>
4006_B00B	UART Control Register 5 (UART1_C5)	8	R/W	00h	<a href="#">40.3.12/976</a>
4006_B00C	UART Extended Data Register (UART1_ED)	8	R	00h	<a href="#">40.3.13/977</a>
4006_B00D	UART Modem Register (UART1_MODEM)	8	R/W	00h	<a href="#">40.3.14/978</a>
4006_B00E	UART Infrared Register (UART1_IR)	8	R/W	00h	<a href="#">40.3.15/979</a>
4006_B010	UART FIFO Parameters (UART1_PFIFO)	8	R/W	See section	<a href="#">40.3.16/980</a>
4006_B011	UART FIFO Control Register (UART1_CFIFO)	8	R/W	00h	<a href="#">40.3.17/981</a>
4006_B012	UART FIFO Status Register (UART1_SFIFO)	8	R/W	C0h	<a href="#">40.3.18/982</a>
4006_B013	UART FIFO Transmit Watermark (UART1_TWFIFO)	8	R/W	00h	<a href="#">40.3.19/983</a>
4006_B014	UART FIFO Transmit Count (UART1_TCFIFO)	8	R	00h	<a href="#">40.3.20/984</a>
4006_B015	UART FIFO Receive Watermark (UART1_RWFIFO)	8	R/W	01h	<a href="#">40.3.21/984</a>
4006_B016	UART FIFO Receive Count (UART1_RCFIFO)	8	R	00h	<a href="#">40.3.22/985</a>

### 40.3.1 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	LBKDIE	RXEDGIE	0	SBR				
Write								
Reset	0	0	0	0	0	0	0	0

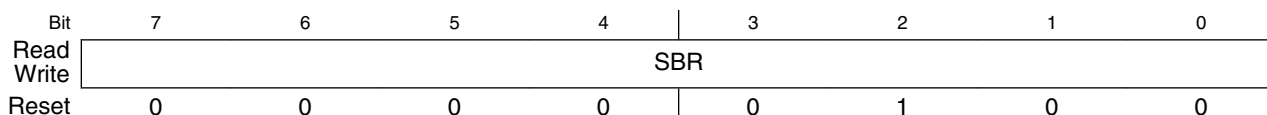
#### UARTx\_BDH field descriptions

Field	Description
7 LBKDIE	<p>LIN Break Detect Interrupt Enable</p> <p>Enables the LIN break detect flag, LBKDIF, to generate interrupt requests based on the state of LBKDDMAS.</p> <p>0 LBKDIF interrupt requests disabled. 1 LBKDIF interrupt requests enabled.</p>
6 RXEDGIE	<p>RxD Input Active Edge Interrupt Enable</p> <p>Enables the receive input active edge, RXEDGIF, to generate interrupt requests.</p> <p>0 Hardware interrupts from RXEDGIF disabled using polling. 1 RXEDGIF interrupt request enabled.</p>
5 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
4–0 SBR	<p>UART Baud Rate Bits</p> <p>The baud rate for the UART is determined by the 13 SBR fields. See <a href="#">Baud rate generation</a> for details.</p> <p><b>NOTE:</b></p> <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> </ul>

### 40.3.2 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



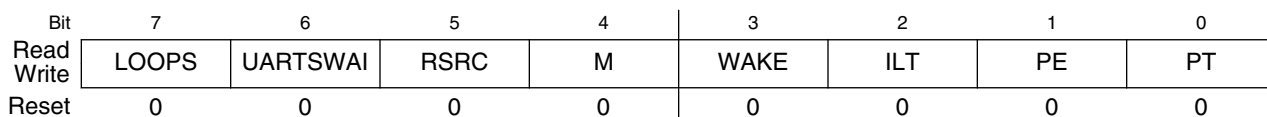
#### UARTx\_BDL field descriptions

Field	Description
7-0 SBR	<p>UART Baud Rate Bits</p> <p>The baud rate for the UART is determined by the 13 SBR fields. See <a href="#">Baud rate generation</a> for details.</p> <p><b>NOTE:</b></p> <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> <li>When the 1/32 narrow pulse width is selected for infrared (IrDA), the baud rate fields must be even, the least significant bit is 0. See MODEM register for more details.</li> </ul>

### 40.3.3 UART Control Register 1 (UARTx\_C1)

This read/write register controls various optional features of the UART system.

Address: Base address + 2h offset



#### UARTx\_C1 field descriptions

Field	Description
7 LOOPS	<p>Loop Mode Select</p> <p>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.</p>

Table continues on the next page...

## UARTx\_C1 field descriptions (continued)

Field	Description
	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: <ul style="list-style-type: none"> <li>Address mark in the most significant bit position of a received data character, or</li> <li>An idle condition on the receive pin input signal.</li> </ul> 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. <b>NOTE:</b> <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> </ul> 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. 0 Parity function disabled. 1 Parity function enabled.
0 PT	Parity Type

Table continues on the next page...

## UARTx\_C1 field descriptions (continued)

Field	Description
	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.

## 40.3.4 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	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Write								
Reset	0	0	0	0	0	0	0	0

## UARTx\_C2 field descriptions

Field	Description
7 TIE	<p>Transmitter Interrupt or DMA Transfer Enable.</p> <p>Enables S1[TDRE] to generate interrupt requests or DMA transfer requests, based on the state of C5[TDMAS].</p> <p><b>NOTE:</b> 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.</p> <p>0 TDRE interrupt and DMA transfer requests disabled. 1 TDRE interrupt or DMA transfer requests enabled.</p>
6 TCIE	<p>Transmission Complete Interrupt Enable</p> <p>Enables the transmission complete flag, S1[TC], to generate interrupt requests .</p> <p>0 TC interrupt requests disabled. 1 TC interrupt requests enabled.</p>
5 RIE	<p>Receiver Full Interrupt or DMA Transfer Enable</p> <p>Enables S1[RDRF] to generate interrupt requests or DMA transfer requests, based on the state of C5[RDMAS].</p> <p>0 RDRF interrupt and DMA transfer requests disabled. 1 RDRF interrupt or DMA transfer requests enabled.</p>
4 ILIE	<p>Idle Line Interrupt Enable</p> <p>Enables the idle line flag, S1[IDLE], to generate interrupt requests</p> <p>0 IDLE interrupt requests disabled. 1 IDLE interrupt requests enabled.</p>

Table continues on the next page...

## UARTx\_C2 field descriptions (continued)

Field	Description
3 TE	<p>Transmitter Enable</p> <p>Enables the UART transmitter. TE can be used to queue an idle preamble by clearing and then setting TE.</p> <p>0 Transmitter off. 1 Transmitter on.</p>
2 RE	<p>Receiver Enable</p> <p>Enables the UART receiver.</p> <p>0 Receiver off. 1 Receiver on.</p>
1 RWU	<p>Receiver Wakeup Control</p> <p>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.</p> <p><b>NOTE:</b> 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.</p> <p>0 Normal operation. 1 RWU enables the wakeup function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU.</p>
0 SBK	<p>Send Break</p> <p> toggling SBK sends one break character from the following: See <a href="#">Transmitting break characters</a> 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 at least 1 clock before assertion of this bit.</p> <ul style="list-style-type: none"> <li>• 10, 11, or 12 logic 0s if S2[BRK13] is cleared</li> <li>• 13 or 14 logic 0s if S2[BRK13] is set.</li> </ul> <p>0 Normal transmitter operation. 1 Queue break characters to be sent.</p>

### 40.3.5 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 as 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	<p>Transmit Data Register Empty Flag</p> <p>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 TDRE 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.</p> <p>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.</p>
6 TC	<p>Transmit Complete Flag</p> <p>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:</p> <ul style="list-style-type: none"> <li>• Writing to D to transmit new data.</li> <li>• Queuing a preamble by clearing and then setting C2[TE].</li> <li>• Queuing a break character by writing 1 to SBK in C2.</li> </ul> <p>0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete).</p>

*Table continues on the next page...*



## UARTx\_S1 field descriptions (continued)

Field	Description
5 RDRF	<p>Receive Data Register Full Flag</p> <p>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.</p> <p>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.</p>
4 IDLE	<p>Idle Line Flag</p> <p>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:</p> <ul style="list-style-type: none"> <li>• 10 consecutive logic 1s if C1[M] = 0</li> <li>• 11 consecutive logic 1s if C1[M] = 1 and C4[M10] = 0</li> <li>• 12 consecutive logic 1s if C1[M] = 1, C4[M10] = 1, and C1[PE] = 1</li> </ul> <p><b>NOTE:</b> 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.</p> <p>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.</p>
3 OR	<p>Receiver Overrun Flag</p> <p>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.</p> <p>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 occurred.</p>
2 NF	<p>Noise Flag</p> <p>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.</p> <p>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.</p>

Table continues on the next page...

**UARTx\_S1 field descriptions (continued)**

Field	Description
1 FE	<p>Framing Error Flag</p> <p>FE is set when a logic 0 is accepted as the stop bit. 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.</p> <p>0 No framing error detected. 1 Framing error.</p>
0 PF	<p>Parity Error Flag</p> <p>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.</p> <p>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.</p>

**40.3.6 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	<p>LIN Break Detect Interrupt Flag</p> <p>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.</p> <p>0 No LIN break character detected. 1 LIN break character detected.</p>
6 RXEDGIF	<p>RxD Pin Active Edge Interrupt Flag</p> <p>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. <a href="#">RXEDGIF description</a></p> <p><b>NOTE:</b> The active edge is detected only in two wire mode and on receiving data coming from the RxD pin.</p> <p>0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.</p>
5 MSBF	<p>Most Significant Bit First</p> <p>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.</p> <p>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].</p>
4 RXINV	<p>Receive Data Inversion</p> <p>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. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity. A zero is represented by a short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity.</p> <p><b>NOTE:</b> Setting RXINV inverts the RxD input for data bits, start and stop bits, break, and idle.</p> <p>0 Receive data is not inverted. 1 Receive data is inverted.</p>
3 RWUID	<p>Receive Wakeup Idle Detect</p> <p>When RWU is set and WAKE is cleared, this field controls whether the idle character that wakes the receiver sets S1[IDLE].</p> <p>0 S1[IDLE] is not set upon detection of an idle character. 1 S1[IDLE] is set upon detection of an idle character.</p>
2 BRK13	<p>Break Transmit Character Length</p> <p>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. <a href="#">Transmitting break characters</a></p>

Table continues on the next page...

### UARTx\_S2 field descriptions (continued)

Field	Description
	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 . <a href="#">Overrun operation</a>  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.

### 40.3.7 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  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.  <b>NOTE:</b> 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

Table continues on the next page...

## UARTx\_C3 field descriptions (continued)

Field	Description
	<p>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.</p> <p>0 TXD pin is an input in single wire mode. 1 TXD pin is an output in single wire mode.</p>
4 TXINV	<p>Transmit Data Inversion.</p> <p>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. In IrDA format, a zero is represented by short high pulse in the middle of a bit time remaining idle low for a one for normal polarity, and a zero is represented by short low pulse in the middle of a bit time remaining idle high for a one for inverted polarity.</p> <p><b>NOTE:</b> 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.</p> <p>0 Transmit data is not inverted. 1 Transmit data is inverted.</p>
3 ORIE	<p>Overrun Error Interrupt Enable</p> <p>Enables the overrun error flag, S1[OR], to generate interrupt requests.</p> <p>0 OR interrupts are disabled. 1 OR interrupt requests are enabled.</p>
2 NEIE	<p>Noise Error Interrupt Enable</p> <p>Enables the noise flag, S1[NF], to generate interrupt requests.</p> <p>0 NF interrupt requests are disabled. 1 NF interrupt requests are enabled.</p>
1 FEIE	<p>Framing Error Interrupt Enable</p> <p>Enables the framing error flag, S1[FE], to generate interrupt requests.</p> <p>0 FE interrupt requests are disabled. 1 FE interrupt requests are enabled.</p>
0 PEIE	<p>Parity Error Interrupt Enable</p> <p>Enables the parity error flag, S1[PF], to generate interrupt requests.</p> <p>0 PF interrupt requests are disabled. 1 PF interrupt requests are enabled.</p>

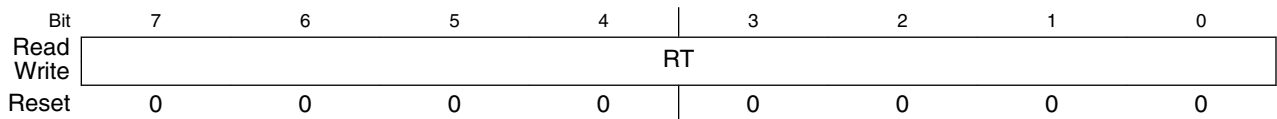
### 40.3.8 UART Data Register (UARTx\_D)

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



**UARTx\_D field descriptions**

Field	Description
7-0 RT	Reads return the contents of the read-only receive data register and writes go to the write-only transmit data register.

### 40.3.9 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	MA							
Write								
Reset	0	0	0	0	0	0	0	0

**UARTx\_MA1 field descriptions**

Field	Description
7-0 MA	Match Address

### 40.3.10 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	MA							
Write								
Reset	0	0	0	0	0	0	0	0

**UARTx\_MA2 field descriptions**

Field	Description
7-0 MA	Match Address

### 40.3.11 UART Control Register 4 (UARTx\_C4)

Address: Base address + Ah offset

Bit	7	6	5	4	3	2	1	0
Read	MAEN1	MAEN2	M10	BRFA				
Write								
Reset	0	0	0	0	0	0	0	0

### UARTx\_C4 field descriptions

Field	Description
7 MAEN1	<p>Match Address Mode Enable 1</p> <p>See <a href="#">Match address operation</a> for more information.</p> <p>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.</p>
6 MAEN2	<p>Match Address Mode Enable 2</p> <p>See <a href="#">Match address operation</a> for more information.</p> <p>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.</p>
5 M10	<p>10-bit Mode select</p> <p>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.</p> <p>See <a href="#">Data format</a> for more information.</p> <p>0 The parity bit is the ninth bit in the serial transmission.                      1 The parity bit is the tenth bit in the serial transmission.</p>
4-0 BRFA	<p>Baud Rate Fine Adjust</p> <p>This bit field is used to add more timing resolution to the average baud frequency, in increments of 1/32. See <a href="#">Baud rate generation</a> for more information.</p>

### 40.3.12 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	<p>Transmitter DMA Select</p> <p>Configures the transmit data register empty flag, S1[TDRE], to generate interrupt or DMA requests if C2[TIE] is set.</p> <p><b>NOTE:</b></p> <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> </ul>

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.  <b>NOTE:</b> If C2[RIE] is cleared, and S1[RDRF] is set, the RDRF DMA and RDRF interrupt request signals are not asserted, regardless of the state of RDMAS.  0 If C2[RIE] and S1[RDRF] are set, the RDRF 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.
3–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

## 40.3.13 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.
5–0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

## 40.3.14 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	0				0	0	0	0
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. <b>NOTE:</b> 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 <a href="#">Hardware flow control</a>
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)(FIFO)
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.

**40.3.15 UART Infrared Register (UARTx\_IR)**

The IR register controls options for setting the infrared configuration.

Address: Base address + Eh offset

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

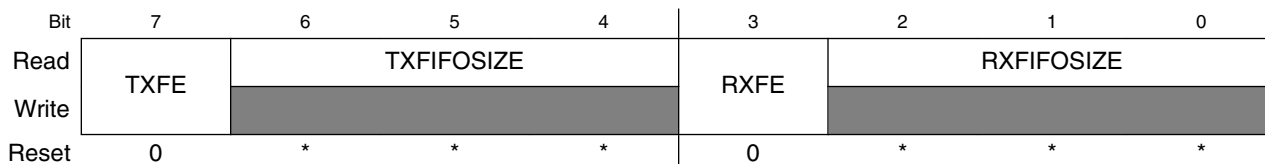
**UARTx\_IR field descriptions**

Field	Description
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 IREN	Infrared enable  Enables/disables the infrared modulation/demodulation.  0 IR disabled. 1 IR enabled.
1–0 TNP	Transmitter narrow pulse  Enables whether the UART transmits a 1/16, 3/16, 1/32, or 1/4 narrow pulse.  00 3/16. 01 1/16. 10 1/32. 11 1/4.

### 40.3.16 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



\* 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	<p>Transmit FIFO Enable</p> <p>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.</p> <p>0 Transmit FIFO is not enabled. Buffer is depth 1. (Legacy support).                      1 Transmit FIFO is enabled. Buffer is depth indicated by TXFIFOSIZE.</p>
6-4 TXFIFOSIZE	<p>Transmit FIFO. Buffer Depth</p> <p>The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read only.</p> <p>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.</p>
3 RXFE	<p>Receive FIFO Enable</p>

Table continues on the next page...

## UARTx\_PFIFO field descriptions (continued)

Field	Description
	<p>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.</p> <p>0 Receive FIFO is not enabled. Buffer is depth 1. (Legacy support)  1 Receive FIFO is enabled. Buffer is depth indicted by RXFIFOSIZE.</p>
2–0 RXFIFOSIZE	<p>Receive FIFO. Buffer Depth</p> <p>The maximum number of receive datawords that can be stored in the receive buffer before an overrun occurs. This field is read only.</p> <p>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.</p>

## 40.3.17 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				
Write	TXFLUSH	RXFLUSH				RXOFE	TXOFE	RXUFE
Reset	0	0	0	0	0	0	0	0

## UARTx\_CFIFO field descriptions

Field	Description
7 TXFLUSH	<p>Transmit FIFO/Buffer Flush</p> <p>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.</p> <p>0 No flush operation occurs.  1 All data in the transmit FIFO/Buffer is cleared out.</p>

Table continues on the next page...

**UARTx\_CFIFO field descriptions (continued)**

Field	Description
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.  0 RXUF flag does not generate an interrupt to the host. 1 RXUF flag generates an interrupt to the host.

**40.3.18 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.

*Table continues on the next page...*

## UARTx\_SFIFO field descriptions (continued)

Field	Description
	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  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.

## 40.3.19 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	TXWATER							
Write	TXWATER							
Reset	0	0	0	0	0	0	0	0

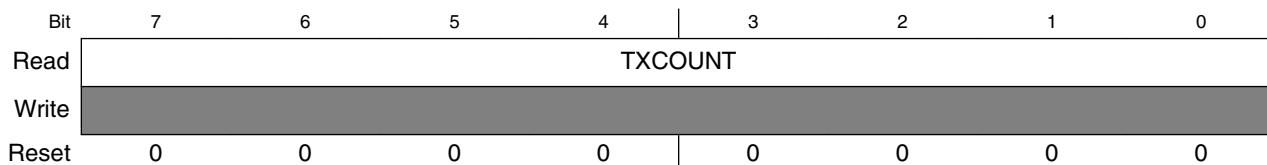
### UARTx\_TWFIFO field descriptions

Field	Description
7-0 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].

### 40.3.20 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



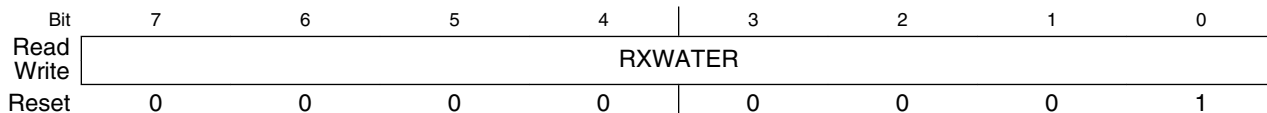
### UARTx\_TCFIFO field descriptions

Field	Description
7-0 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.

### 40.3.21 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





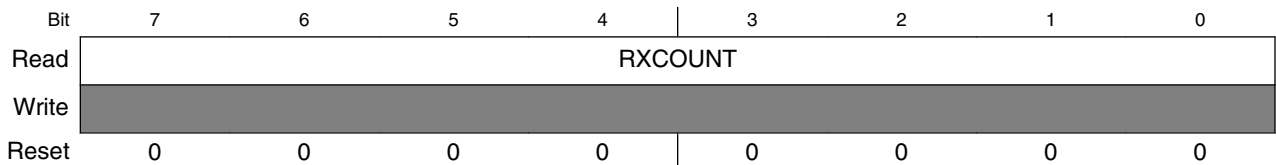
**UARTx\_RWFIFO field descriptions**

Field	Description
7–0 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.

**40.3.22 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

**UARTx\_RCFIFO field descriptions**

Field	Description
7–0 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.

**40.4 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.

## 40.4.1 Transmitter

Figure 40-67. Transmitter Block Diagram

### 40.4.1.1 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).

### 40.4.1.2 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.

### 40.4.1.3 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 programming sequences.

Setting C2[TE] automatically loads the transmit shift register with the following preamble:

- 10 logic 1s if C1[M] = 0
- 11 logic 1s if C1[M] = 1 and C4[M10] = 0
- 12 logic 1s if C1[M] = 1, C4[M10] = 1, C1[PE] = 1

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.

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.

#### 40.4.1.4 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] and C4[M10]. See the following table.

**Table 40-72. Transmit break character length**

S2[BRK13]	C1[M]	C4[M10]	C1[PE]	Bits transmitted
0	0	—	—	10
0	1	1	0	11
0	1	1	1	12
1	0	—	—	13
1	1	—	—	14

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.

### 40.4.1.5 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] 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.

### 40.4.1.6 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.

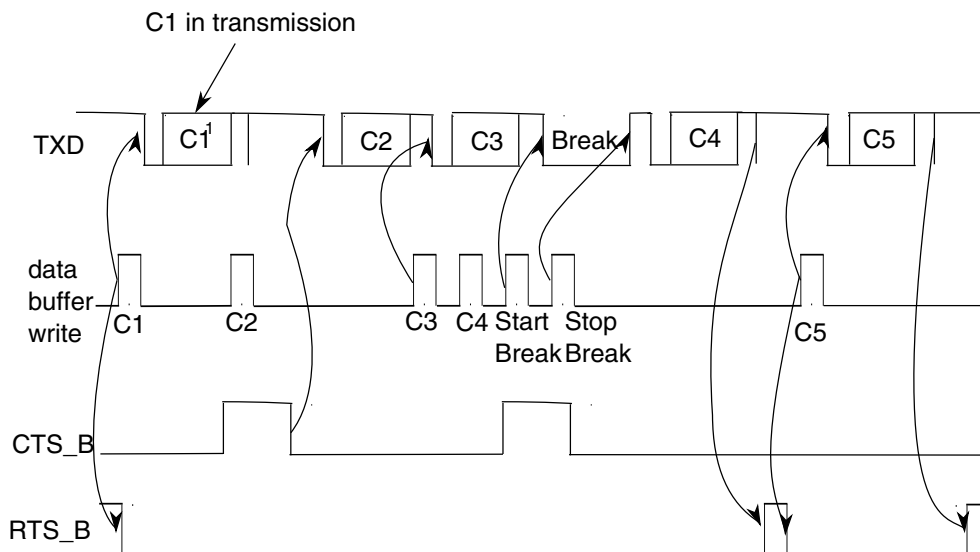
The transmitter's CTS signal can also be enabled even if the same UART receiver's RTS signal is disabled.

#### 40.4.1.7 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 40-68. Transmitter RTS and CTS timing diagram**

## 40.4.2 Receiver

**Figure 40-69. UART receiver block diagram**

### 40.4.2.1 Receiver character length

The UART receiver can accommodate 8-, 9-, or 10-bit data characters. The states of C1[M], C1[PE] and C4[M10] determine the length of data characters. When receiving 9 or 10-bit data, C3[R8] is the ninth bit (bit 8).

### 40.4.2.2 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].



**Table 40-73. 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 40-74. 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).

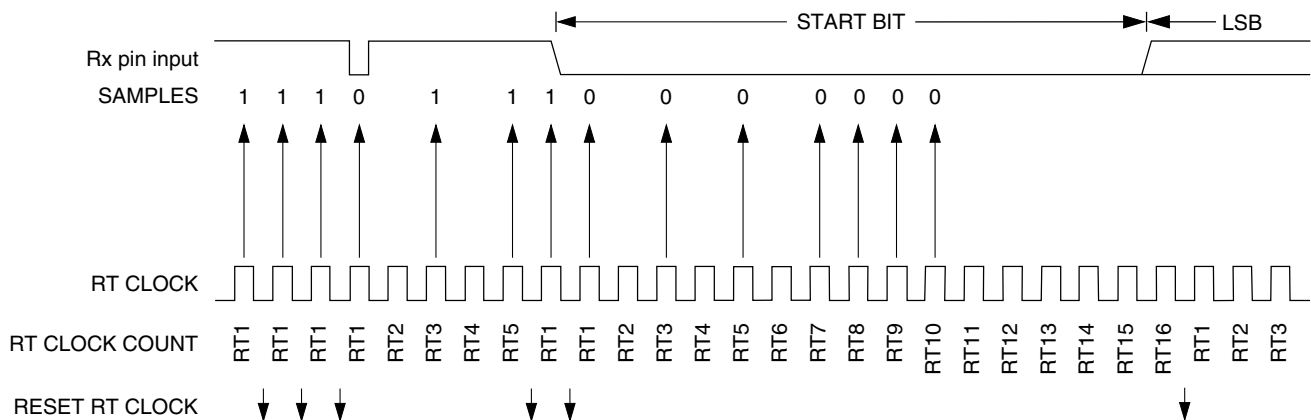
To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. The following table summarizes the results of the stop bit samples.



**Table 40-75. Stop bit recovery**

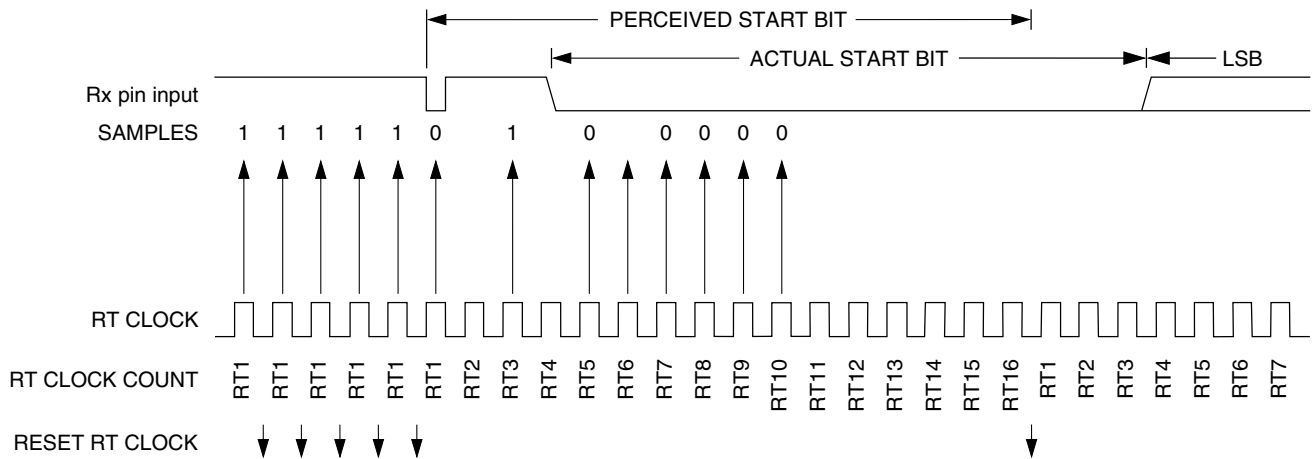
RT8, RT9, and RT10 samples	Framing error flag	Noise flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

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 40-71. Start bit search example 1**

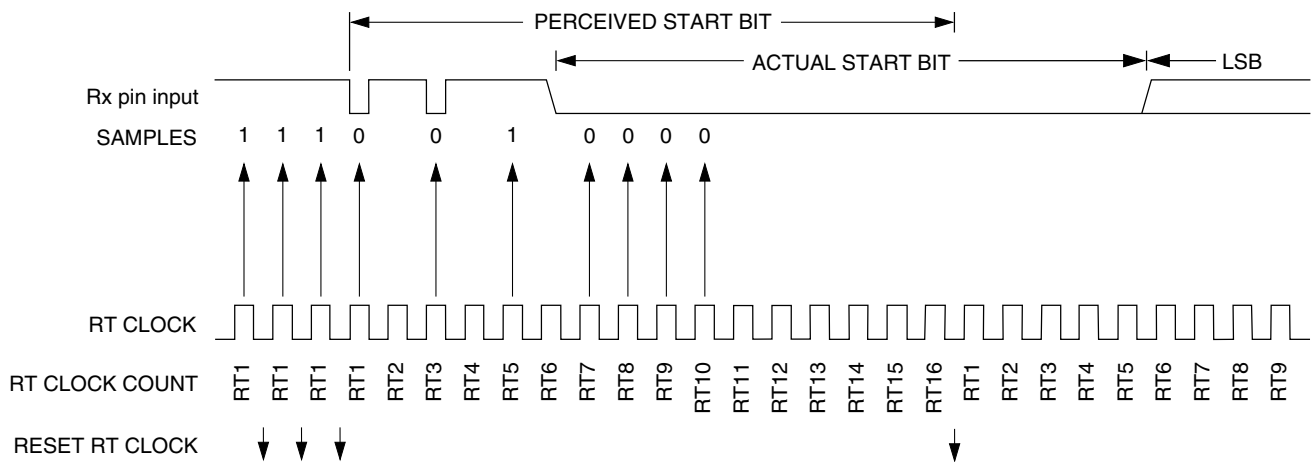
In the following figure, verification sample at RT3 is high. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

**Functional description**



**Figure 40-72. 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 40-73. 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.



## Functional description

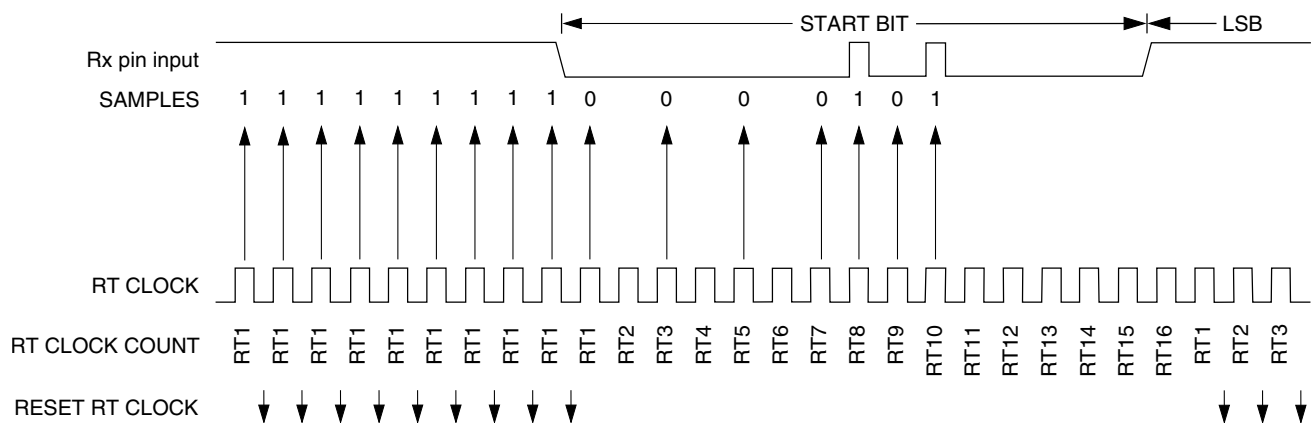


Figure 40-76. Start bit search example 6

### 40.4.2.5 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.

### 40.4.2.6 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 40-76. Receive break character detection threshold**

LBKDE	M	M10	PE	Threshold (bits)
0	0	—	—	10
0	1	0	—	11
0	1	1	1	12
1	0	—	—	11
1	1	—	—	12

While S2[LBKDE] is set, it will have these effects on the UART registers:

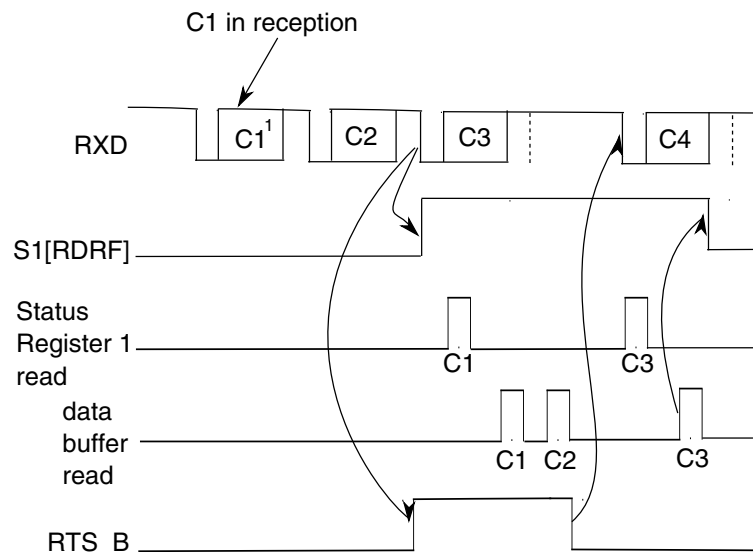
- Prevents S1[RDRF], S1[FE], S1[NF], and S1[PF] from being set. However, if they are already set, they will remain set.
- Sets the LIN break detect interrupt flag, S2[LBKDIF], if a LIN break character is received.

#### 40.4.2.7 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 40-77. Receiver hardware flow control timing diagram**

### 40.4.2.8 Infrared decoder

The infrared decoder converts the received character from the IrDA format to the NRZ format used by the receiver. It also has a 16-RT clock counter that filters noise and indicates when a 1 is received.

#### 40.4.2.8.1 Start bit detection

When S2[RXINV] is cleared, the first rising edge of the received character corresponds to the start bit. The infrared decoder resets its counter. At this time, the receiver also begins its start bit detection process. After the start bit is detected, the receiver synchronizes its bit times to this start bit time. For the rest of the character reception, the infrared decoder's counter and the receiver's bit time counter count independently from each other.

#### 40.4.2.8.2 Noise filtering

Any further rising edges detected during the first half of the infrared decoder counter are ignored by the decoder. Any pulses less than one RT clocks can be undetected by it regardless of whether it is seen in the first or second half of the count.

#### 40.4.2.8.3 Low-bit detection

During the second half of the decoder count, a rising edge is decoded as a 0, which is sent to the receiver. The decoder counter is also reset.

#### 40.4.2.8.4 High-bit detection

At 16-RT clocks after the previous rising edge, if a rising edge is not seen, then the decoder sends a 1 to the receiver.

If the next bit is a 0, which arrives late, then a low-bit is detected according to [Low-bit detection](#). The value sent to the receiver is changed from 1 to a 0. Then, if a noise pulse occurs outside the receiver's bit time sampling period, then the delay of a 0 is not recorded as noise.

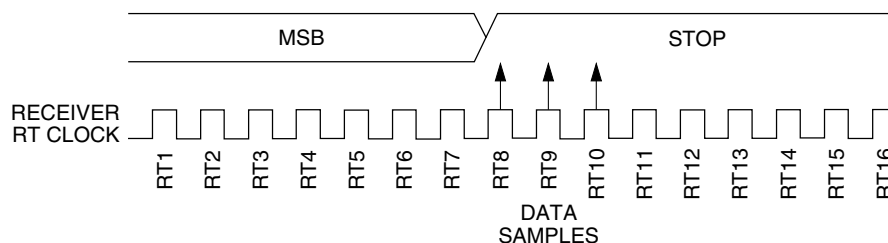
#### 40.4.2.9 Baud rate tolerance

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.

##### 40.4.2.9.1 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 40-78. Slow data**

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles (9 bit times  $\times$  16 RT cycles + 10 RT cycles).

## Functional description

With the misaligned character shown in the [Figure 40-78](#), the receiver counts 154 RT cycles at the point when the count of the transmitting device is 147 RT cycles (9 bit times  $\times$  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  $\times$  16 RT cycles + 10 RT cycles).

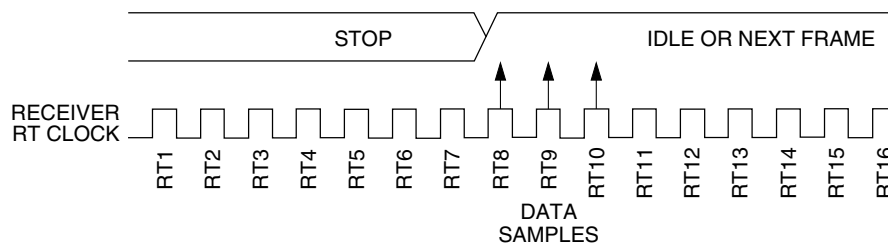
With the misaligned character shown in the [Figure 40-78](#), the receiver counts 170 RT cycles at the point when the count of the transmitting device is 163 RT cycles (10 bit times  $\times$  16 RT cycles + 3 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$((170 - 163) \div 170) \times 100 = 4.12\%$$

### 40.4.2.9.2 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 40-79. Fast data**

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles (9 bit times  $\times$  16 RT cycles + 10 RT cycles).

With the misaligned character shown in the [Figure 40-79](#), the receiver counts 154 RT cycles at the point when the count of the transmitting device is 160 RT cycles (10 bit times  $\times$  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  $\times$  16 RT cycles + 10 RT cycles).

With the misaligned character shown in the [Figure 40-79](#), the receiver counts 170 RT cycles at the point when the count of the transmitting device is 176 RT cycles (11 bit times  $\times$  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:

$$((170 - 176) \div 170) \times 100 = 3.53\%$$

### 40.4.2.10 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.

#### 40.4.2.10.1 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.

#### 40.4.2.10.2 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 preceding 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.

#### **40.4.2.10.3 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.

### 40.4.3 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 40-77](#) lists the available baud divisor fine adjust values.

$$\text{UART baud rate} = \text{UART module clock} / (16 \times (\text{SBR}[12:0] + \text{BRFD}))$$

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 40-77. 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
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 40-78. Baud rate fine adjust**

BRFA	Baud Rate Fractional Divisor (BRFD)
00000	0/32 = 0
00001	1/32 = 0.03125
00010	2/32 = 0.0625
00011	3/32 = 0.09375
00100	4/32 = 0.125
00101	5/32 = 0.15625
00110	6/32 = 0.1875
00111	7/32 = 0.21875
01000	8/32 = 0.25
01001	9/32 = 0.28125
01010	10/32 = 0.3125
01011	11/32 = 0.34375
01100	12/32 = 0.375
01101	13/32 = 0.40625
01110	14/32 = 0.4375
01111	15/32 = 0.46875
10000	16/32 = 0.5
10001	17/32 = 0.53125
10010	18/32 = 0.5625
10011	19/32 = 0.59375
10100	20/32 = 0.625
10101	21/32 = 0.65625
10110	22/32 = 0.6875
10111	23/32 = 0.71875
11000	24/32 = 0.75
11001	25/32 = 0.78125
11010	26/32 = 0.8125
11011	27/32 = 0.84375
11100	28/32 = 0.875
11101	29/32 = 0.90625
11110	30/32 = 0.9375
11111	31/32 = 0.96875

#### 40.4.4 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] and C4[M10].

### 40.4.4.1 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. 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 40-79. 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 <sup>1</sup>	0	1
1	1	7	0	1	1

1. The address bit identifies the frame as an address character. See [Receiver wakeup](#).

### 40.4.4.2 Nine-bit configuration

When C1[M] is set and C4[M10] 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, 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 40-80. 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 <a href="#">Eight-bit configuration</a>				

*Table continues on the next page...*

**Table 40-80. Configuration of 9-bit data formats (continued)**

C1[PE]	UC1[M]	C1[M10]	Start bit	Data bits	Address bits	Parity bits	Stop bit	
0	0	1	Invalid configuration					
0	1	0	1	9	0	0	1	
0	1	0	1	8	1 <sup>1</sup>	0	1	
0	1	1	Invalid Configuration					
1	0	0	See <a href="#">Eight-bit configuration</a>					
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 <sup>2</sup>	1	1	

1. The address bit identifies the frame as an address character.
2. The address bit identifies the frame as an address character.

**Note**

Unless in 9-bit mode with M10 set, do not use address mark wakeup with parity enabled.

**40.4.4.3 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.

**40.4.4.3.1 Eight-bit format with parity disabled**

The most significant bit can be used for address mark wakeup.



**Figure 40-80. Eight bits of data with LSB first**



**Figure 40-81. Eight bits of data with MSB first**

#### 40.4.4.3.2 Eight-bit format with parity enabled



**Figure 40-82. Seven bits of data with LSB first and parity**



**Figure 40-83. Seven bits of data with MSB first and parity**

#### 40.4.4.3.3 Nine-bit format with parity disabled

The most significant bit can be used for address mark wakeup.



**Figure 40-84. Nine bits of data with LSB first**

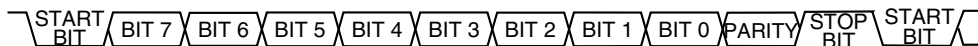


**Figure 40-85. Nine bits of data with MSB first**

#### 40.4.4.3.4 Nine-bit format with parity enabled



**Figure 40-86. Eight bits of data with LSB first and parity**



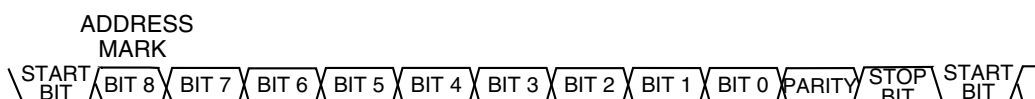
**Figure 40-87. Eight bits of data with MSB first and parity**

#### 40.4.4.3.5 Non-memory mapped tenth bit for parity

The most significant memory-mapped bit can be used for address mark wakeup.



**Figure 40-88. Nine bits of data with LSB first and parity**



**Figure 40-89. Nine bits of data with MSB first and parity**

### 40.4.5 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.

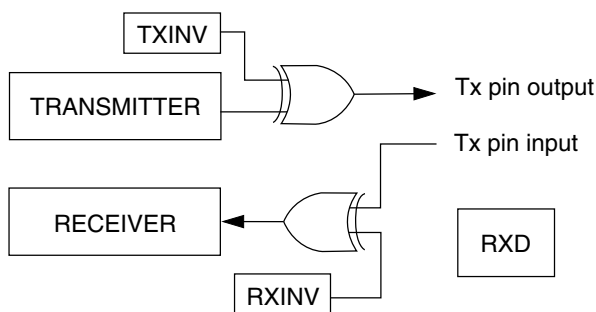


Figure 40-90. Single-wire operation (C1[LOOPS] = 1, C1[RSRC] = 1)

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).

### 40.4.6 Loop operation

In loop operation, the transmitter output goes to the receiver input. The unsynchronized receiver input signal is disconnected from the UART.

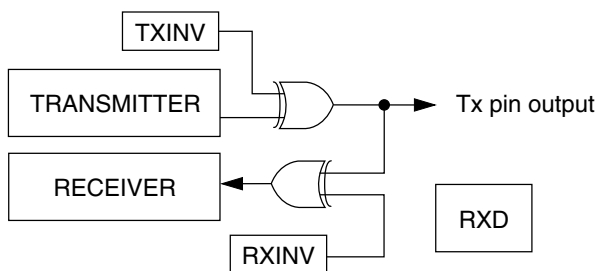


Figure 40-91. Loop operation (C1[LOOPS] = 1, C1[RSRC] = 0)

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).



## 40.4.7 Infrared interface

The UART provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses and transforming them to serial bits, which are sent to the UART. The IrDA physical layer specification defines a half-duplex infrared communication link for exchanging data. The full standard includes data rates up to 16 Mbits/s. This design covers data rates only between 2.4 kbits/s and 115.2 kbits/s.

The UART has an infrared transmit encoder and receive decoder. The UART transmits serial bits of data that are encoded by the infrared submodule to transmit a narrow pulse for every zero bit. No pulse is transmitted for every one bit. When receiving data, the IR pulses are detected using an IR photo diode and transformed to CMOS levels by the IR receive decoder, external from the MCU. The narrow pulses are then stretched by the infrared receive decoder to get back to a serial bit stream to be received by the UART. The polarity of transmitted pulses and expected receive pulses can be inverted so that a direct connection can be made to external IrDA transceiver modules that use active low pulses.

The infrared submodule receives its clock sources from the UART. One of these two clocks are selected in the infrared submodule to generate either 3/16, 1/16, 1/32, or 1/4 narrow pulses during transmission.

### 40.4.7.1 Infrared transmit encoder

The infrared transmit encoder converts serial bits of data from transmit shift register to the TXD signal. A narrow pulse is transmitted for a zero bit and no pulse for a one bit. The narrow pulse is sent in the middle of the bit with a duration of 1/32, 1/16, 3/16, or 1/4 of a bit time. A narrow high pulse is transmitted for a zero bit when C3[TXINV] is cleared, while a narrow low pulse is transmitted for a zero bit when C3[TXINV] is set.

### 40.4.7.2 Infrared receive decoder

The infrared receive block converts data from the RXD signal to the receive shift register. A narrow pulse is expected for each zero received and no pulse is expected for each one received. A narrow high pulse is expected for a zero bit when S2[RXINV] is cleared, while a narrow low pulse is expected for a zero bit when S2[RXINV] is set. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

## 40.5 Reset

All registers reset to a particular value are indicated in [Memory map and registers](#).

## 40.6 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 40-81. 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	-

### 40.6.1 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.

### 40.6.1.1 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.

### 40.6.1.2 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.

### 40.6.1.3 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).

## 40.7 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 40-82. 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.

## 40.8 Application information

This section describes the UART application information.

### 40.8.1 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.
  - 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.

**40.8.2 Overrun (OR) flag implications**

To be flexible, the overrun flag (OR) operates slightly 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.

**40.8.2.1 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.
2. Clear S1[OR].

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.

### 40.8.3 Match address registers

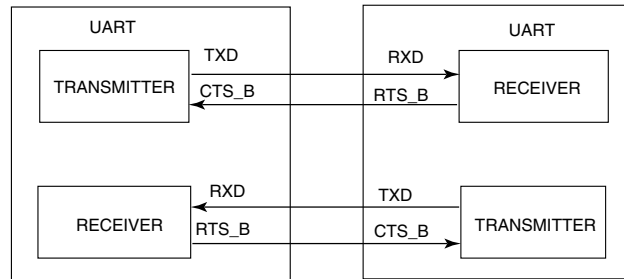
The two match address registers allow a second match address function for a broadcast or general call address to the serial bus, as an example.

### 40.8.4 Modem feature

This section describes the modem features.

#### 40.8.4.1 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.

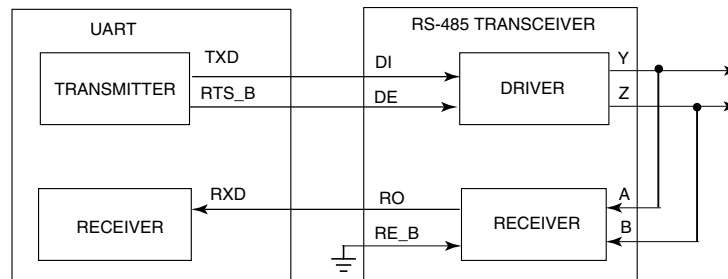


**Figure 40-92. Ready-to-receive**

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.

#### 40.8.4.2 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 40-93. 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.

#### 40.8.5 IrDA minimum pulse width

The IrDA specifies a minimum pulse width of 1.6  $\mu\text{s}$ . The UART hardware does not include a mechanism to restrict/force the pulse width to be greater than or equal to 1.6  $\mu\text{s}$ . However, configuring the baud rate to 115.2 kbit/s and the narrow pulse width to 3/16 of a bit time results in a pulse width of 1.6  $\mu\text{s}$ .

## 40.8.6 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 41

## General-Purpose Input/Output (GPIO)

### 41.1 Introduction

#### NOTE

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

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.

#### 41.1.1 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

#### NOTE

The GPIO module is clocked by system clock.

#### 41.1.2 Modes of operation

The following table depicts different modes of operation and the behavior of the GPIO module in these modes.

**Table 41-1. 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.

### 41.1.3 GPIO signal descriptions

**Table 41-2. 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.

### 41.1.3.1 Detailed signal description

**Table 41-3. GPIO interface-detailed signal descriptions**

Signal	I/O	Description	
PORTA31–PORTA0 PORTB31–PORTB0 PORTC31–PORTC0 PORTD31–PORTD0 PORTE31–PORTE0	I/O	General-purpose input/output	
		State meaning	Asserted: The pin is logic 1. Deasserted: The pin is logic 0.
		Timing	Assertion: 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.  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.

## 41.2 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.

#### 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	<a href="#">41.2.1/1021</a>
400F_F004	Port Set Output Register (GPIOA_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.2/1022</a>
400F_F008	Port Clear Output Register (GPIOA_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.3/1022</a>
400F_F00C	Port Toggle Output Register (GPIOA_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.4/1023</a>

*Table continues on the next page...*

## GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
400F_F010	Port Data Input Register (GPIOA_PDIR)	32	R	0000_0000h	<a href="#">41.2.5/1023</a>
400F_F014	Port Data Direction Register (GPIOA_PDDR)	32	R/W	0000_0000h	<a href="#">41.2.6/1024</a>
400F_F040	Port Data Output Register (GPIOB_PDOR)	32	R/W	0000_0000h	<a href="#">41.2.1/1021</a>
400F_F044	Port Set Output Register (GPIOB_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.2/1022</a>
400F_F048	Port Clear Output Register (GPIOB_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.3/1022</a>
400F_F04C	Port Toggle Output Register (GPIOB_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.4/1023</a>
400F_F050	Port Data Input Register (GPIOB_PDIR)	32	R	0000_0000h	<a href="#">41.2.5/1023</a>
400F_F054	Port Data Direction Register (GPIOB_PDDR)	32	R/W	0000_0000h	<a href="#">41.2.6/1024</a>
400F_F080	Port Data Output Register (GPIOC_PDOR)	32	R/W	0000_0000h	<a href="#">41.2.1/1021</a>
400F_F084	Port Set Output Register (GPIOC_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.2/1022</a>
400F_F088	Port Clear Output Register (GPIOC_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.3/1022</a>
400F_F08C	Port Toggle Output Register (GPIOC_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.4/1023</a>
400F_F090	Port Data Input Register (GPIOC_PDIR)	32	R	0000_0000h	<a href="#">41.2.5/1023</a>
400F_F094	Port Data Direction Register (GPIOC_PDDR)	32	R/W	0000_0000h	<a href="#">41.2.6/1024</a>
400F_F0C0	Port Data Output Register (GPIOD_PDOR)	32	R/W	0000_0000h	<a href="#">41.2.1/1021</a>
400F_F0C4	Port Set Output Register (GPIOD_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.2/1022</a>
400F_F0C8	Port Clear Output Register (GPIOD_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.3/1022</a>
400F_F0CC	Port Toggle Output Register (GPIOD_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.4/1023</a>
400F_F0D0	Port Data Input Register (GPIOD_PDIR)	32	R	0000_0000h	<a href="#">41.2.5/1023</a>
400F_F0D4	Port Data Direction Register (GPIOD_PDDR)	32	R/W	0000_0000h	<a href="#">41.2.6/1024</a>
400F_F100	Port Data Output Register (GPIOE_PDOR)	32	R/W	0000_0000h	<a href="#">41.2.1/1021</a>
400F_F104	Port Set Output Register (GPIOE_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.2/1022</a>

Table continues on the next page...

## GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
400F_F108	Port Clear Output Register (GPIOE_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.3/1022</a>
400F_F10C	Port Toggle Output Register (GPIOE_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">41.2.4/1023</a>
400F_F110	Port Data Input Register (GPIOE_PDIR)	32	R	0000_0000h	<a href="#">41.2.5/1023</a>
400F_F114	Port Data Direction Register (GPIOE_PDDR)	32	R/W	0000_0000h	<a href="#">41.2.6/1024</a>

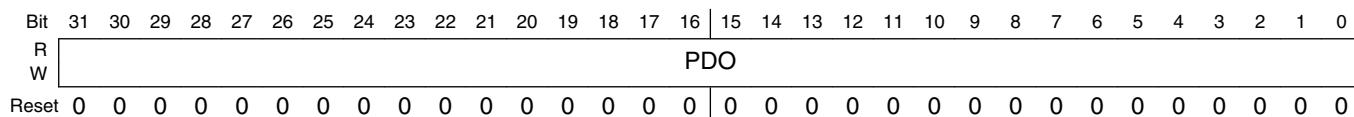
### 41.2.1 Port Data Output Register (GPIOx\_PDOR)

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



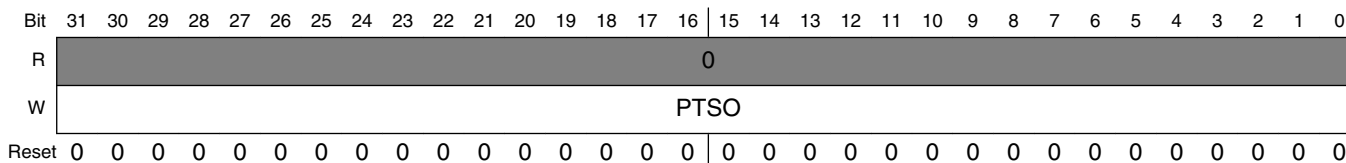
#### GPIOx\_PDOR field descriptions

Field	Description
31–0 PDO	<p>Port Data Output</p> <p>Register bits for unbonded pins return a undefined value when read.</p> <p>0 Logic level 0 is driven on pin, provided pin is configured for general-purpose output.</p> <p>1 Logic level 1 is driven on pin, provided pin is configured for general-purpose output.</p>

### 41.2.2 Port Set Output Register (GPIOx\_PSOR)

This register configures whether to set the fields of the PDOR.

Address: Base address + 4h offset



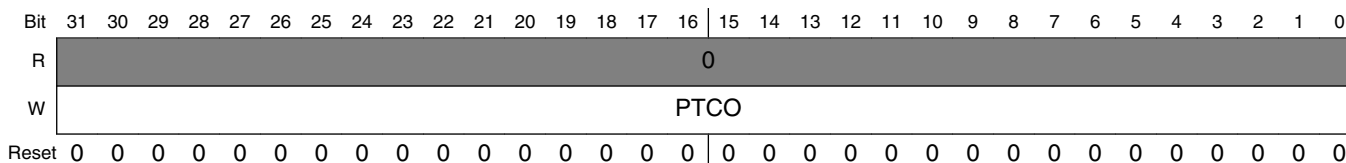
#### GPIOx\_PSOR field descriptions

Field	Description
31–0 PTSO	<p>Port Set Output</p> <p>Writing to this register will update the contents of the corresponding bit in the PDOR as follows:</p> <p>0 Corresponding bit in PDORn does not change.</p> <p>1 Corresponding bit in PDORn is set to logic 1.</p>

### 41.2.3 Port Clear Output Register (GPIOx\_PCOR)

This register configures whether to clear the fields of PDOR.

Address: Base address + 8h offset

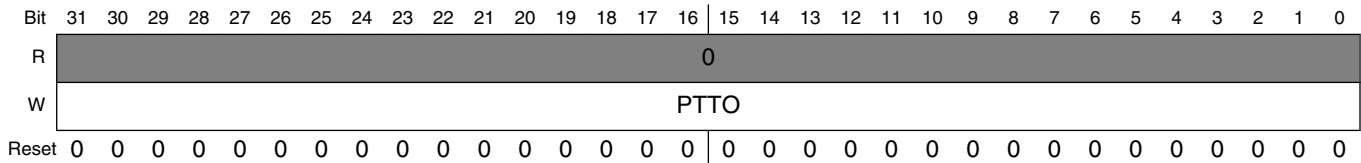


#### GPIOx\_PCOR field descriptions

Field	Description
31–0 PTCO	<p>Port Clear Output</p> <p>Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows:</p> <p>0 Corresponding bit in PDORn does not change.</p> <p>1 Corresponding bit in PDORn is cleared to logic 0.</p>

## 41.2.4 Port Toggle Output Register (GPIOx\_PTOR)

Address: Base address + Ch offset



### GPIOx\_PTOR field descriptions

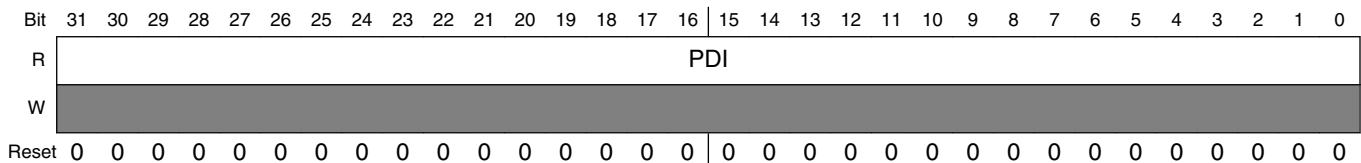
Field	Description
31–0 PTTO	<p>Port Toggle Output</p> <p>Writing to this register will update the contents of the corresponding bit in the PDOR as follows:</p> <p>0 Corresponding bit in PDORn does not change.</p> <p>1 Corresponding bit in PDORn is set to the inverse of its existing logic state.</p>

## 41.2.5 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



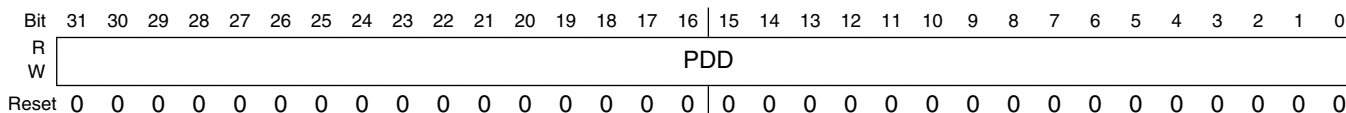
### GPIOx\_PDIR field descriptions

Field	Description
31–0 PDI	<p>Port Data Input</p> <p>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.</p> <p>0 Pin logic level is logic 0, or is not configured for use by digital function.</p> <p>1 Pin logic level is logic 1.</p>

## 41.2.6 Port Data Direction Register (GPIOx\_PDDR)

The PDDR configures the individual port pins for input or output.

Address: Base address + 14h offset



### GPIOx\_PDDR field descriptions

Field	Description
31–0 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.

## 41.3 Functional description

### 41.3.1 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.

### 41.3.2 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 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.



# Chapter 42

## JTAG Controller (JTAGC)

### 42.1 Introduction

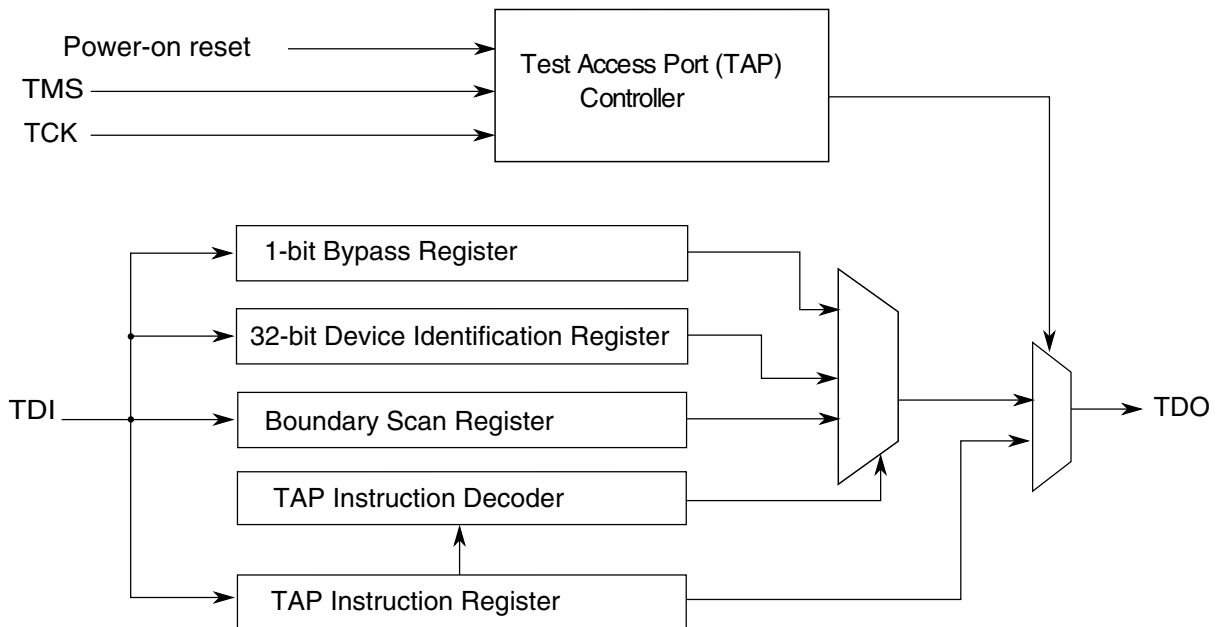
#### NOTE

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

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

#### 42.1.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as [Register description](#) for more information about the JTAGC registers.



**Figure 42-1. JTAG (IEEE 1149.1) block diagram**

## 42.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to [Table 42-3](#) for a list of supported instructions.
- Bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

## 42.1.3 Modes of operation

The JTAGC block uses a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

### 42.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

### 42.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in [JTAGC block instructions](#).

### 42.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

## 42.2 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

**Table 42-1. JTAG signal properties**

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z <sup>1</sup>	—
TMS	Input	Test Mode Select	—	Up

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

### 42.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

### 42.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

### 42.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in [TAP controller state machine](#).

### 42.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

## 42.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

### 42.3.1 Instruction register

The JTAGC block uses a 4-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 0001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	3	2	1	0
R	0	0	0	1
W	Instruction Code			
Reset:	0	0	0	1

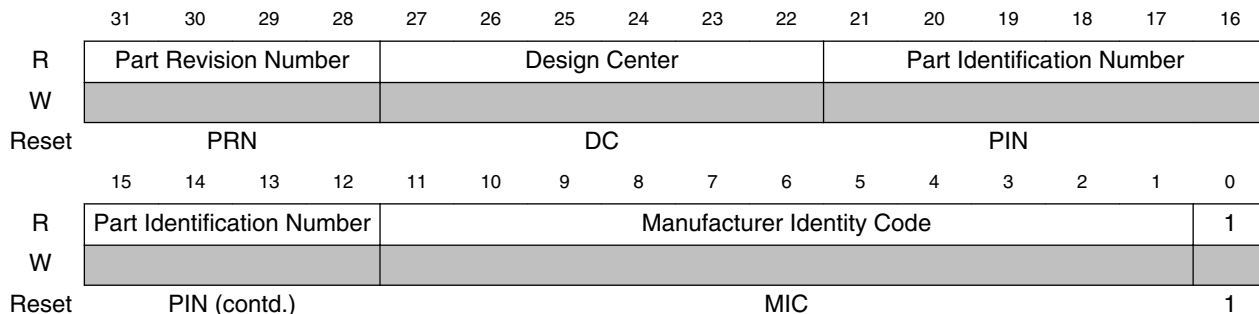
**Figure 42-2. Instruction register**

### 42.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

### 42.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.



The following table describes the device identification register functions.

**Table 42-2. Device identification register field descriptions**

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is On this device, the PIN mirrors bits 9-0 of the SIM_SDID[REVID] field. Please see the <a href="#">SIM_SDID register</a> description for more detail.
DC	Design Center. Indicates the design center. Value is 0x2C.
PIN	Part Identification Number. Contains the part number of the device. On this device, the PIN mirrors bits 9-0 of the SIM_SDID register. Please see the <a href="#">SIM_SDID register</a> description for more detail.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

### 42.3.4 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in [Boundary scan](#). The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.



## 42.4 Functional description

This section explains the JTAGC functional description.

### 42.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

### 42.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction.

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

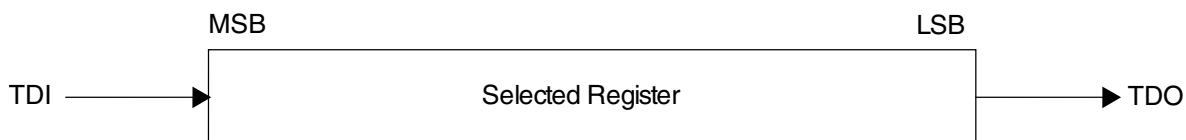
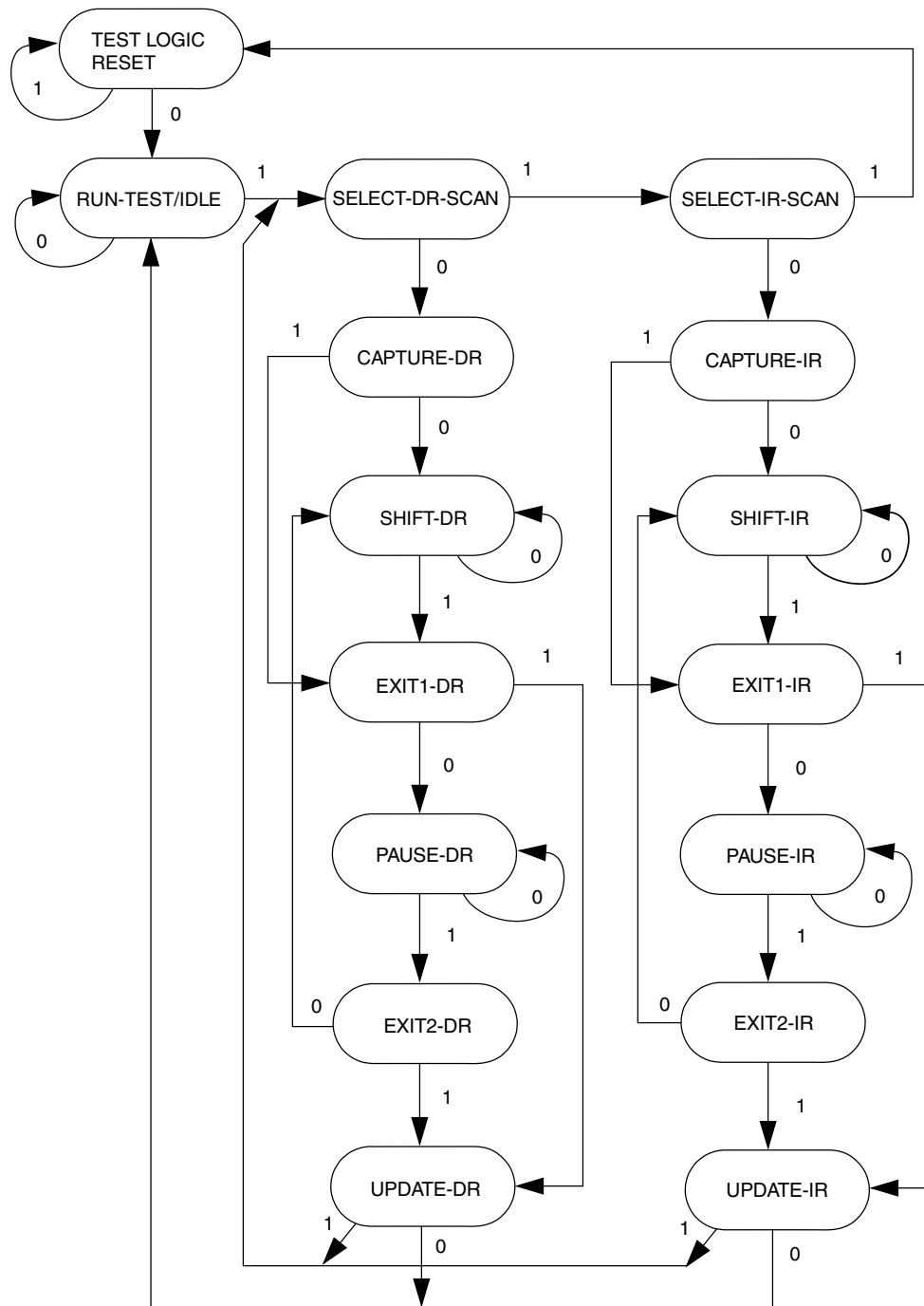


Figure 42-3. Shifting data through a register

### 42.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

**Figure 42-4. IEEE 1149.1-2001 TAP controller finite state machine**

### 42.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting the JTAGC enable to a logic 1 value.

### 42.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

### 42.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

**Table 42-3. 4-bit JTAG instructions**

Instruction	Code[3:0]	Instruction summary
IDCODE	0000	Selects device identification register for shift
SAMPLE/PRELOAD	0010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	0011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	0100	Selects boundary scan register and applies preloaded values to output pins. <b>NOTE:</b> Execution of this instruction asserts functional reset.
Factory debug reserved	0101	Intended for factory debug only
Factory debug reserved	0110	Intended for factory debug only
Factory debug reserved	0111	Intended for factory debug only
ARM JTAG-DP Reserved	1000	This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information.
HIGHZ	1001	Selects bypass register and three-states all output pins. <b>NOTE:</b> Execution of this instruction asserts functional reset.
ARM JTAG-DP Reserved	1010	This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information.

*Table continues on the next page...*

**Table 42-3. 4-bit JTAG instructions (continued)**

Instruction	Code[3:0]	Instruction summary
ARM JTAG-DP Reserved	1011	This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information.
CLAMP	1100	Selects bypass register and applies preloaded values to output pins.  <b>NOTE:</b> Execution of this instruction asserts functional reset.
ARM JTAG-DP Reserved	1110	This instruction goes the ARM JTAG-DP controller. See the ARM JTAG-DP documentation for more information.
BYPASS	1111	Selects bypass register for data operations

#### 42.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

#### 42.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

### 42.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

### 42.4.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

### 42.4.4.5 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

### 42.4.4.6 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

#### **42.4.4.7 BYPASS instruction**

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

#### **42.4.5 Boundary scan**

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

### **42.5 Initialization/Application information**

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Place the JTAGC in reset through TAP controller state machine transitions controlled by TMS
2. Load the appropriate instruction for the test or action to be performed

---

**How to Reach Us:**

**Home Page:**

[freescale.com](http://freescale.com)

**Web Support:**

[freescale.com/support](http://freescale.com/support)

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [freescale.com/SalesTermsandConditions](http://freescale.com/SalesTermsandConditions).

Freescale, the Freescale logo, and Kinetis are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. ARM and Cortex are the registered trademarks of ARM Limited. © 2014 Freescale Semiconductor, Inc.

Document Number: K02P64M100SFARM  
Rev. 1  
7/2014

