

---

# KL13 Sub-Family Reference Manual

Supports:MKL13Z32VLH4, MKL13Z64VLH4, MKL13Z32VLK4,  
MKL13Z64VLK4

Document Number: KL13P80M48SF3RM  
Rev. 2.1, 07/2016





# Contents

Section number	Title	Page
----------------	-------	------

## Chapter 1 About This Document

1.1	Overview.....	35
1.1.1	Purpose.....	35
1.1.2	Audience.....	35
1.2	Conventions.....	35
1.2.1	Numbering systems.....	35
1.2.2	Typographic notation.....	36
1.2.3	Special terms.....	36

## Chapter 2 Introduction

2.1	Overview.....	37
2.1.1	Sub-family introduction.....	37
2.2	Module functional categories.....	38
2.2.1	ARM Cortex-M0+ core modules.....	39
2.2.2	System modules.....	39
2.2.3	Memories and memory interfaces.....	40
2.2.4	Clocks.....	40
2.2.5	Security and integrity modules.....	41
2.2.6	Analog modules.....	41
2.2.7	Timer modules.....	41
2.2.8	Communication interfaces.....	42
2.2.9	Human-machine interfaces.....	42
2.3	Module to module interconnects.....	43
2.3.1	Interconnection overview.....	43
2.3.2	Analog reference options.....	45

## Chapter 3 Core Overview

Section number	Title	Page
3.1	ARM Cortex-M0+ core introduction.....	47
3.1.1	Buses, interconnects, and interfaces.....	47
3.1.2	System tick timer.....	47
3.1.3	Debug facilities.....	47
3.1.4	Core privilege levels.....	48
3.2	Nested vectored interrupt controller (NVIC) .....	48
3.2.1	Interrupt priority levels.....	48
3.2.2	Non-maskable interrupt.....	48
3.2.3	Interrupt channel assignments.....	48
3.3	AWIC introduction.....	51
3.3.1	Wake-up sources.....	51

## Chapter 4 Memory Map

4.1	Introduction.....	53
4.2	Flash memory.....	53
4.2.1	Flash memory map.....	53
4.2.2	Flash security.....	54
4.2.3	Flash modes.....	54
4.2.4	Erase all flash contents.....	54
4.2.5	FTFA_FOFT register.....	54
4.3	SRAM.....	55
4.3.1	SRAM sizes.....	55
4.3.2	SRAM ranges.....	55
4.3.3	SRAM retention in low power modes.....	56
4.4	System Register file.....	56
4.5	System memory map.....	57
4.6	Bit Manipulation Engine.....	57
4.7	Peripheral bridge (AIPS-Lite) memory map.....	58
4.7.1	Read-after-write sequence and required serialization of memory operations.....	58

Section number	Title	Page
4.7.2	Peripheral bridge (AIPS-Lite) memory map.....	59

## Chapter 5 Clock Distribution

5.1	Introduction.....	63
5.2	Programming model.....	63
5.3	High-level device clocking diagram.....	63
5.4	Clock definitions.....	64
5.4.1	Device clock summary.....	65
5.5	Internal clocking requirements.....	67
5.5.1	Clock divider values after reset.....	67
5.5.2	VLPR mode clocking.....	68
5.6	Clock gating.....	68
5.7	Module clocks.....	69
5.7.1	PMC 1-kHz LPO clock.....	70
5.7.2	COP clocking.....	70
5.7.3	RTC clocking.....	70
5.7.4	RTC_CLKOUT and CLKOUT32K clocking.....	71
5.7.5	LPTMR clocking.....	72
5.7.6	TPM clocking.....	73
5.7.7	LPUART clocking.....	73
5.7.8	FlexIO clocking.....	74

## Chapter 6 Reset and Boot

6.1	Introduction.....	75
6.2	Reset.....	75
6.2.1	Power-on reset (POR).....	76
6.2.2	System reset sources.....	76
6.2.3	MCU resets.....	79
6.2.4	RESET pin .....	80

Section number	Title	Page
6.3	Boot.....	80
6.3.1	Boot sources.....	81
6.3.2	FOPT boot options.....	81
6.3.3	Boot sequence.....	83

## Chapter 7 Power Management

7.1	Introduction.....	85
7.2	Clocking modes.....	85
7.2.1	Partial Stop.....	85
7.2.2	DMA Wakeup.....	86
7.2.3	Compute Operation.....	87
7.2.4	Peripheral Doze.....	88
7.2.5	Clock gating.....	89
7.3	Power modes.....	89
7.4	Entering and exiting power modes.....	91
7.5	Module operation in low-power modes.....	92

## Chapter 8 Security

8.1	Introduction.....	97
8.1.1	Flash security.....	97
8.1.2	Security interactions with other modules.....	97

## Chapter 9 Debug

9.1	Introduction.....	99
9.2	Debug port pin descriptions.....	99
9.3	SWD status and control registers.....	100
9.3.1	MDM-AP Control Register.....	101
9.3.2	MDM-AP Status Register.....	102
9.4	Debug resets.....	104
9.5	Micro Trace Buffer (MTB).....	104

<b>Section number</b>	<b>Title</b>	<b>Page</b>
9.6	Debug in low-power modes.....	105
9.7	Debug and security.....	106

## **Chapter 10 Pinouts and Packaging**

10.1	Introduction.....	107
10.2	Signal multiplexing integration.....	107
10.2.1	I/O Port control and interrupt module features.....	108
10.2.2	Clock gating.....	108
10.2.3	Signal multiplexing constraints.....	108
10.3	KL13 Signal Multiplexing and Pin Assignments.....	108
10.4	KL13 Family Pinouts.....	111
10.5	Module Signal Description Tables.....	116
10.5.1	Core modules.....	116
10.5.2	System modules.....	116
10.5.3	Clock modules.....	117
10.5.4	Memories and memory interfaces.....	117
10.5.5	Analog.....	117
10.5.6	Timer Modules.....	118
10.5.7	Communication interfaces.....	119
10.5.8	Human-machine interfaces (HMI).....	121

## **Chapter 11 Port Control and Interrupts (PORT)**

11.1	Chip-specific PORT information.....	123
11.2	Port control and interrupt summary.....	124
11.3	Introduction.....	125
11.4	Overview.....	125
11.4.1	Features.....	125
11.4.2	Modes of operation.....	126
11.5	External signal description.....	127

Section number	Title	Page
11.6	Detailed signal description.....	127
11.7	Memory map and register definition.....	127
11.7.1	Pin Control Register n (PORTx_PCRn).....	133
11.7.2	Global Pin Control Low Register (PORTx_GPCLR).....	136
11.7.3	Global Pin Control High Register (PORTx_GPCHR).....	136
11.7.4	Interrupt Status Flag Register (PORTx_ISFR).....	137
11.8	Functional description.....	137
11.8.1	Pin control.....	137
11.8.2	Global pin control.....	138
11.8.3	External interrupts.....	138

## Chapter 12 System Integration Module (SIM)

12.1	Chip-specific SIM information.....	141
12.1.1	COP clocks.....	141
12.2	Introduction.....	141
12.2.1	Features.....	141
12.3	Memory map and register definition.....	142
12.3.1	System Options Register 1 (SIM_SOPT1).....	143
12.3.2	System Options Register 2 (SIM_SOPT2).....	144
12.3.3	System Options Register 4 (SIM_SOPT4).....	146
12.3.4	System Options Register 5 (SIM_SOPT5).....	147
12.3.5	System Options Register 7 (SIM_SOPT7).....	149
12.3.6	System Device Identification Register (SIM_SDID).....	150
12.3.7	System Clock Gating Control Register 4 (SIM_SCGC4).....	151
12.3.8	System Clock Gating Control Register 5 (SIM_SCGC5).....	153
12.3.9	System Clock Gating Control Register 6 (SIM_SCGC6).....	155
12.3.10	System Clock Gating Control Register 7 (SIM_SCGC7).....	157
12.3.11	System Clock Divider Register 1 (SIM_CLKDIV1).....	158
12.3.12	Flash Configuration Register 1 (SIM_FCFG1).....	159



Section number	Title	Page
12.3.13	Flash Configuration Register 2 (SIM_FCFG2).....	161
12.3.14	Unique Identification Register Mid-High (SIM_UIDMH).....	162
12.3.15	Unique Identification Register Mid Low (SIM_UIDML).....	162
12.3.16	Unique Identification Register Low (SIM_UIDL).....	163
12.3.17	COP Control Register (SIM_COPC).....	163
12.3.18	Service COP (SIM_SRV COP).....	165
12.4	Functional description.....	165
12.4.1	COP watchdog operation.....	165

## Chapter 13 Kinetic ROM Bootloader

13.1	Chip-Specific Information.....	169
13.2	Introduction.....	169
13.3	Functional Description.....	171
13.3.1	Memory Maps.....	171
13.3.2	The Kinetic Bootloader Configuration Area (BCA).....	172
13.3.3	Start-up Process.....	173
13.3.4	Clock Configuration.....	175
13.3.5	Bootloader Entry Point.....	176
13.3.6	Bootloader Protocol.....	177
13.3.7	Bootloader Packet Types.....	180
13.3.8	Bootloader Command API.....	186
13.3.9	Bootloader Exit state.....	199
13.4	Peripherals Supported.....	199
13.4.1	I2C Peripheral.....	199
13.4.2	SPI Peripheral.....	201
13.4.3	LPUART Peripheral.....	203
13.5	Get/SetProperty Command Properties.....	206
13.5.1	Property Definitions.....	207
13.6	Kinetic Bootloader Status Error Codes.....	208

Section number	Title	Page
<b>Chapter 14</b>		
<b>System Mode Controller (SMC)</b>		
14.1	Chip-specific SMC information.....	211
14.2	Introduction.....	211
14.3	Modes of operation.....	211
14.4	Memory map and register descriptions.....	213
14.4.1	Power Mode Protection register (SMC_PMPROT).....	214
14.4.2	Power Mode Control register (SMC_PMCTRL).....	215
14.4.3	Stop Control Register (SMC_STOPCTRL).....	217
14.4.4	Power Mode Status register (SMC_PMSTAT).....	218
14.5	Functional description.....	219
14.5.1	Power mode transitions.....	219
14.5.2	Power mode entry/exit sequencing.....	222
14.5.3	Run modes.....	224
14.5.4	Wait modes.....	226
14.5.5	Stop modes.....	227
14.5.6	Debug in low power modes.....	230
<b>Chapter 15</b>		
<b>Power Management Controller (PMC)</b>		
15.1	Introduction.....	233
15.2	Features.....	233
15.3	Low-voltage detect (LVD) system.....	233
15.3.1	LVD reset operation.....	234
15.3.2	LVD interrupt operation.....	234
15.3.3	Low-voltage warning (LVW) interrupt operation.....	234
15.4	I/O retention.....	235
15.5	Memory map and register descriptions.....	235
15.5.1	Low Voltage Detect Status And Control 1 register (PMC_LVDSC1).....	236
15.5.2	Low Voltage Detect Status And Control 2 register (PMC_LVDSC2).....	237

Section number	Title	Page
15.5.3	Regulator Status And Control register (PMC_REGSC).....	238

## Chapter 16 Miscellaneous Control Module (MCM)

16.1	Introduction.....	241
16.1.1	Features.....	241
16.2	Memory map/register descriptions.....	241
16.2.1	Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC).....	242
16.2.2	Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC).....	242
16.2.3	Platform Control Register (MCM_PLACR).....	243
16.2.4	Compute Operation Control Register (MCM_CPO).....	246

## Chapter 17 Crossbar Switch Lite (AXBS-Lite)

17.1	Chip-specific AXBS-Lite information.....	249
17.1.1	Crossbar-light switch master assignments.....	249
17.1.2	Crossbar switch slave assignments.....	249
17.2	Introduction.....	249
17.2.1	Features.....	250
17.3	Memory Map / Register Definition.....	250
17.4	Functional Description.....	250
17.4.1	General operation.....	250
17.4.2	Arbitration.....	251
17.5	Initialization/application information.....	253

## Chapter 18 Low-Leakage Wakeup Unit (LLWU)

18.1	Chip-specific LLWU information.....	255
18.1.1	LLWU interrupt.....	255
18.1.2	Wake-up Sources.....	255
18.2	Introduction.....	256
18.2.1	Features.....	256
18.2.2	Modes of operation.....	257

Section number	Title	Page
18.2.3	Block diagram.....	258
18.3	LLWU signal descriptions.....	259
18.4	Memory map/register definition.....	259
18.4.1	LLWU Pin Enable 1 register (LLWU_PE1).....	260
18.4.2	LLWU Pin Enable 2 register (LLWU_PE2).....	261
18.4.3	LLWU Pin Enable 3 register (LLWU_PE3).....	262
18.4.4	LLWU Pin Enable 4 register (LLWU_PE4).....	263
18.4.5	LLWU Module Enable register (LLWU_ME).....	264
18.4.6	LLWU Flag 1 register (LLWU_F1).....	266
18.4.7	LLWU Flag 2 register (LLWU_F2).....	268
18.4.8	LLWU Flag 3 register (LLWU_F3).....	269
18.4.9	LLWU Pin Filter 1 register (LLWU_FILT1).....	271
18.4.10	LLWU Pin Filter 2 register (LLWU_FILT2).....	272
18.5	Functional description.....	273
18.5.1	LLS mode.....	274
18.5.2	VLLS modes.....	274
18.5.3	Initialization.....	274

## Chapter 19 Peripheral Bridge (AIPS-Lite)

19.1	Chip-specific AIPS-Lite information.....	275
19.1.1	Number of peripheral bridges.....	275
19.1.2	Memory maps.....	275
19.2	Introduction.....	275
19.2.1	Features.....	275
19.2.2	General operation.....	276
19.3	Memory map/register definition.....	276
19.3.1	Master Privilege Register A (AIPS_MPRA).....	277
19.3.2	Peripheral Access Control Register (AIPS_PACR <sub>n</sub> ).....	279
19.3.3	Peripheral Access Control Register (AIPS <sub>n</sub> ).....	0

Section number	Title	Page
19.4	Functional description.....	283
19.4.1	Access support.....	283

## Chapter 20 Direct Memory Access Multiplexer (DMAMUX)

20.1	Chip-specific DMAMUX information.....	285
20.1.1	DMA MUX Request Sources.....	285
20.1.2	DMA transfers via PIT trigger.....	287
20.2	Introduction.....	287
20.2.1	Overview.....	287
20.2.2	Features.....	288
20.2.3	Modes of operation.....	288
20.3	External signal description.....	289
20.4	Memory map/register definition.....	289
20.4.1	Channel Configuration register (DMAMUX <sub>x</sub> _CHCFG <sub>n</sub> ).....	289
20.5	Functional description.....	290
20.5.1	DMA channels with periodic triggering capability.....	291
20.5.2	DMA channels with no triggering capability.....	293
20.5.3	Always-enabled DMA sources.....	293
20.6	Initialization/application information.....	295
20.6.1	Reset.....	295
20.6.2	Enabling and configuring sources.....	295

## Chapter 21 DMA Controller Module

21.1	Introduction.....	299
21.1.1	Overview.....	299
21.1.2	Features.....	300
21.2	DMA Transfer Overview.....	301
21.3	Memory Map/Register Definition.....	302
21.3.1	Source Address Register (DMA_SAR <sub>n</sub> ).....	303

Section number	Title	Page
21.3.2	Destination Address Register (DMA_DAR <sub>n</sub> ).....	304
21.3.3	DMA Status Register / Byte Count Register (DMA_DSR_BCR <sub>n</sub> ).....	305
21.3.4	DMA Control Register (DMA_DCR <sub>n</sub> ).....	307
21.4	Functional Description.....	311
21.4.1	Transfer requests (Cycle-Steal and Continuous modes).....	311
21.4.2	Channel initialization and startup.....	312
21.4.3	Dual-Address Data Transfer Mode.....	313
21.4.4	Advanced Data Transfer Controls: Auto-Alignment.....	314
21.4.5	Termination.....	315

## Chapter 22 Reset Control Module (RCM)

22.1	Introduction.....	317
22.2	Reset memory map and register descriptions.....	317
22.2.1	System Reset Status Register 0 (RCM_SRS0).....	318
22.2.2	System Reset Status Register 1 (RCM_SRS1).....	319
22.2.3	Reset Pin Filter Control register (RCM_RPFC).....	320
22.2.4	Reset Pin Filter Width register (RCM_RPFW).....	321
22.2.5	Force Mode Register (RCM_FM).....	323
22.2.6	Mode Register (RCM_MR).....	323
22.2.7	Sticky System Reset Status Register 0 (RCM_SSRS0).....	324
22.2.8	Sticky System Reset Status Register 1 (RCM_SSRS1).....	325

## Chapter 23 Analog-to-Digital Converter (ADC)

23.1	Chip-specific ADC information.....	327
23.1.1	ADC instantiation information.....	327
23.1.2	DMA Support on ADC.....	327
23.1.3	ADC0 connections/channel assignment.....	327
23.1.4	ADC analog supply and reference connections.....	329
23.1.5	Alternate clock.....	329

Section number	Title	Page
23.2	Introduction.....	329
23.2.1	Features.....	330
23.2.2	Block diagram.....	330
23.3	ADC signal descriptions.....	331
23.3.1	Analog Power (VDDA).....	332
23.3.2	Analog Ground (VSSA).....	332
23.3.3	Voltage Reference Select.....	332
23.3.4	Analog Channel Inputs (ADx).....	333
23.3.5	Differential Analog Channel Inputs (DADx).....	333
23.4	Memory map and register definitions.....	333
23.4.1	ADC Status and Control Registers 1 (ADCx_SC1n).....	334
23.4.2	ADC Configuration Register 1 (ADCx_CFG1).....	338
23.4.3	ADC Configuration Register 2 (ADCx_CFG2).....	339
23.4.4	ADC Data Result Register (ADCx_Rn).....	340
23.4.5	Compare Value Registers (ADCx_CVn).....	342
23.4.6	Status and Control Register 2 (ADCx_SC2).....	343
23.4.7	Status and Control Register 3 (ADCx_SC3).....	345
23.4.8	ADC Offset Correction Register (ADCx_OFS).....	346
23.4.9	ADC Plus-Side Gain Register (ADCx_PG).....	347
23.4.10	ADC Minus-Side Gain Register (ADCx_MG).....	347
23.4.11	ADC Plus-Side General Calibration Value Register (ADCx_CLPD).....	348
23.4.12	ADC Plus-Side General Calibration Value Register (ADCx_CLPS).....	349
23.4.13	ADC Plus-Side General Calibration Value Register (ADCx_CLP4).....	349
23.4.14	ADC Plus-Side General Calibration Value Register (ADCx_CLP3).....	350
23.4.15	ADC Plus-Side General Calibration Value Register (ADCx_CLP2).....	350
23.4.16	ADC Plus-Side General Calibration Value Register (ADCx_CLP1).....	351
23.4.17	ADC Plus-Side General Calibration Value Register (ADCx_CLP0).....	351
23.4.18	ADC Minus-Side General Calibration Value Register (ADCx_CLMD).....	352
23.4.19	ADC Minus-Side General Calibration Value Register (ADCx_CLMS).....	352

Section number	Title	Page
23.4.20	ADC Minus-Side General Calibration Value Register (ADCx_CLM4).....	353
23.4.21	ADC Minus-Side General Calibration Value Register (ADCx_CLM3).....	353
23.4.22	ADC Minus-Side General Calibration Value Register (ADCx_CLM2).....	354
23.4.23	ADC Minus-Side General Calibration Value Register (ADCx_CLM1).....	354
23.4.24	ADC Minus-Side General Calibration Value Register (ADCx_CLM0).....	355
23.5	Functional description.....	355
23.5.1	Clock select and divide control.....	356
23.5.2	Voltage reference selection.....	357
23.5.3	Hardware trigger and channel selects.....	357
23.5.4	Conversion control.....	358
23.5.5	Automatic compare function.....	366
23.5.6	Calibration function.....	367
23.5.7	User-defined offset function.....	369
23.5.8	Temperature sensor.....	370
23.5.9	MCU wait mode operation.....	371
23.5.10	MCU Normal Stop mode operation.....	371
23.6	Initialization information.....	372
23.6.1	ADC module initialization example.....	373
23.7	Application information.....	375
23.7.1	External pins and routing.....	375
23.7.2	Sources of error.....	377

## Chapter 24 Comparator (CMP)

24.1	Chip-specific CMP information.....	381
24.1.1	CMP instantiation information.....	381
24.1.2	CMP input connections.....	381
24.1.3	CMP external references.....	382
24.1.4	CMP trigger mode.....	382
24.2	Introduction.....	383



Section number	Title	Page
24.2.1	CMP features.....	383
24.2.2	6-bit DAC key features.....	384
24.2.3	ANMUX key features.....	384
24.2.4	CMP, DAC and ANMUX diagram.....	384
24.2.5	CMP block diagram.....	385
24.3	Memory map/register definitions.....	387
24.3.1	CMP Control Register 0 (CMP <sub>x</sub> _CR0).....	387
24.3.2	CMP Control Register 1 (CMP <sub>x</sub> _CR1).....	388
24.3.3	CMP Filter Period Register (CMP <sub>x</sub> _FPR).....	389
24.3.4	CMP Status and Control Register (CMP <sub>x</sub> _SCR).....	390
24.3.5	DAC Control Register (CMP <sub>x</sub> _DACCR).....	391
24.3.6	MUX Control Register (CMP <sub>x</sub> _MUXCR).....	391
24.4	Functional description.....	392
24.4.1	CMP functional modes.....	393
24.4.2	Power modes.....	396
24.4.3	Startup and operation.....	397
24.4.4	Low-pass filter.....	398
24.5	CMP interrupts.....	400
24.6	DMA support.....	400
24.7	CMP Asynchronous DMA support.....	400
24.8	Digital-to-analog converter.....	401
24.9	DAC functional description.....	401
24.9.1	Voltage reference source select.....	401
24.10	DAC resets.....	402
24.11	DAC clocks.....	402
24.12	DAC interrupts.....	402
24.13	CMP Trigger Mode.....	402

**Chapter 25**  
**12-bit Digital-to-Analog Converter (DAC)**

Section number	Title	Page
25.1	Introduction.....	403
25.2	Features.....	403
25.3	Block diagram.....	403
25.4	Memory map/register definition.....	404
25.4.1	DAC Data Low Register (DACx_DATnL).....	405
25.4.2	DAC Data High Register (DACx_DATnH).....	405
25.4.3	DAC Status Register (DACx_SR).....	406
25.4.4	DAC Control Register (DACx_C0).....	407
25.4.5	DAC Control Register 1 (DACx_C1).....	408
25.4.6	DAC Control Register 2 (DACx_C2).....	408
25.5	Functional description.....	409
25.5.1	DAC data buffer operation.....	409
25.5.2	DMA operation.....	411
25.5.3	Resets.....	411
25.5.4	Low-Power mode operation.....	411

## Chapter 26 Voltage Reference (VREFV1)

26.1	Introduction.....	413
26.1.1	Overview.....	414
26.1.2	Features.....	414
26.1.3	Modes of Operation.....	414
26.1.4	VREF Signal Descriptions.....	415
26.2	Memory Map and Register Definition.....	415
26.2.1	VREF Trim Register (VREF_TRM).....	416
26.2.2	VREF Status and Control Register (VREF_SC).....	417
26.3	Functional Description.....	418
26.3.1	Voltage Reference Disabled, SC[VREFEN] = 0.....	418
26.3.2	Voltage Reference Enabled, SC[VREFEN] = 1.....	418
26.4	Internal voltage regulator.....	420

Section number	Title	Page
26.5	Initialization/Application Information.....	420

## Chapter 27 Multipurpose Clock Generator Lite (MCG\_Lite)

27.1	Introduction .....	423
27.1.1	Features .....	423
27.1.2	Block diagram .....	423
27.2	Memory map and register definition.....	424
27.2.1	MCG Control Register 1 (MCG_C1).....	425
27.2.2	MCG Control Register 2 (MCG_C2).....	426
27.2.3	MCG Status Register (MCG_S).....	427
27.2.4	MCG Status and Control Register (MCG_SC).....	427
27.2.5	MCG Miscellaneous Control Register (MCG_MC).....	428
27.3	Functional description.....	429
27.3.1	Clock mode switching .....	429
27.3.2	LIRC divider 1 .....	430
27.3.3	LIRC divider 2 .....	430
27.3.4	Enable LIRC in Stop mode .....	430
27.3.5	MCG-Lite in Low-power mode .....	431

## Chapter 28 Oscillator (OSC)

28.1	Chip-specific OSC information.....	433
28.1.1	OSC modes of operation with MCG_Lite and RTC.....	433
28.2	Introduction.....	433
28.3	Features and Modes.....	433
28.4	Block Diagram.....	434
28.5	OSC Signal Descriptions.....	434
28.6	External Crystal / Resonator Connections.....	435
28.7	External Clock Connections.....	437
28.8	Memory Map/Register Definitions.....	437

Section number	Title	Page
28.8.1	OSC Memory Map/Register Definition.....	438
28.9	Functional Description.....	439
28.9.1	OSC module states.....	439
28.9.2	OSC module modes.....	441
28.9.3	Counter.....	443
28.9.4	Reference clock pin requirements.....	443
28.10	Reset.....	443
28.11	Low power modes operation.....	444
28.12	Interrupts.....	444

## Chapter 29 Timer/PWM Module (TPM)

29.1	Chip-specific TPM information.....	445
29.1.1	TPM instantiation information.....	445
29.1.2	Clock options.....	446
29.1.3	Trigger options.....	446
29.1.4	Global timebase.....	447
29.1.5	TPM interrupts.....	447
29.2	Introduction.....	448
29.2.1	TPM Philosophy.....	448
29.2.2	Features.....	448
29.2.3	Modes of operation.....	449
29.2.4	Block diagram.....	449
29.3	TPM Signal Descriptions.....	450
29.3.1	TPM_EXTCLK — TPM External Clock.....	450
29.3.2	TPM_CHn — TPM Channel (n) I/O Pin.....	451
29.4	Memory Map and Register Definition.....	451
29.4.1	Status and Control (TPMx_SC).....	453
29.4.2	Counter (TPMx_CNT).....	454
29.4.3	Modulo (TPMx_MOD).....	455

<b>Section number</b>	<b>Title</b>	<b>Page</b>
29.4.4	Channel (n) Status and Control (TPM <sub>x</sub> _CnSC).....	456
29.4.5	Channel (n) Value (TPM <sub>x</sub> _CnV).....	458
29.4.6	Capture and Compare Status (TPM <sub>x</sub> _STATUS).....	458
29.4.7	Channel Polarity (TPM <sub>x</sub> _POL).....	460
29.4.8	Configuration (TPM <sub>x</sub> _CONF).....	461
29.5	Functional description.....	464
29.5.1	Clock domains.....	464
29.5.2	Prescaler.....	465
29.5.3	Counter.....	465
29.5.4	Input Capture Mode.....	468
29.5.5	Output Compare Mode.....	469
29.5.6	Edge-Aligned PWM (EPWM) Mode.....	470
29.5.7	Center-Aligned PWM (CPWM) Mode.....	472
29.5.8	Registers Updated from Write Buffers.....	474
29.5.9	DMA.....	474
29.5.10	Output triggers.....	475
29.5.11	Reset Overview.....	475
29.5.12	TPM Interrupts.....	476

## **Chapter 30**

### **Periodic Interrupt Timer (PIT)**

30.1	Chip-specific PIT information.....	477
30.1.1	PIT/DMA periodic trigger assignments .....	477
30.1.2	PIT/ADC triggers.....	477
30.1.3	PIT/TPM triggers.....	477
30.1.4	PIT/DAC triggers.....	477
30.2	Introduction.....	478
30.2.1	Block diagram.....	478
30.2.2	Features.....	478
30.3	Signal description.....	479

Section number	Title	Page
30.4	Memory map/register description.....	479
30.4.1	PIT Module Control Register (PIT_MCR).....	480
30.4.2	PIT Upper Lifetime Timer Register (PIT_LTMR64H).....	481
30.4.3	PIT Lower Lifetime Timer Register (PIT_LTMR64L).....	481
30.4.4	Timer Load Value Register (PIT_LDVAL $n$ ).....	482
30.4.5	Current Timer Value Register (PIT_CVAL $n$ ).....	482
30.4.6	Timer Control Register (PIT_TCTRL $n$ ).....	483
30.4.7	Timer Flag Register (PIT_TFLG $n$ ).....	484
30.5	Functional description.....	484
30.5.1	General operation.....	484
30.5.2	Interrupts.....	486
30.5.3	Chained timers.....	486
30.6	Initialization and application information.....	486
30.7	Example configuration for chained timers.....	487
30.8	Example configuration for the lifetime timer.....	488

## Chapter 31 Low-Power Timer (LPTMR)

31.1	Chip-specific LPTMR information.....	491
31.1.1	LPTMR instantiation information.....	491
31.1.2	LPTMR pulse counter input options.....	491
31.1.3	LPTMR prescaler/glitch filter clocking options.....	491
31.2	Introduction.....	492
31.2.1	Features.....	492
31.2.2	Modes of operation.....	493
31.3	LPTMR signal descriptions.....	493
31.3.1	Detailed signal descriptions.....	493
31.4	Memory map and register definition.....	494
31.4.1	Low Power Timer Control Status Register (LPTMR $x$ _CSR).....	494
31.4.2	Low Power Timer Prescale Register (LPTMR $x$ _PSR).....	495

Section number	Title	Page
31.4.3	Low Power Timer Compare Register (LPTMR <sub>x</sub> _CMR).....	497
31.4.4	Low Power Timer Counter Register (LPTMR <sub>x</sub> _CNR).....	497
31.5	Functional description.....	498
31.5.1	LPTMR power and reset.....	498
31.5.2	LPTMR clocking.....	498
31.5.3	LPTMR prescaler/glitch filter.....	498
31.5.4	LPTMR compare.....	500
31.5.5	LPTMR counter.....	500
31.5.6	LPTMR hardware trigger.....	501
31.5.7	LPTMR interrupt.....	501

## Chapter 32 Real Time Clock (RTC)

32.1	Chip-specific RTC information.....	503
32.1.1	RTC Instantiation Information.....	503
32.1.2	RTC_CLKOUT options.....	503
32.2	Introduction.....	503
32.2.1	Features.....	503
32.2.2	Modes of operation.....	504
32.2.3	RTC signal descriptions.....	504
32.3	Register definition.....	504
32.3.1	RTC Time Seconds Register (RTC_TSR).....	505
32.3.2	RTC Time Prescaler Register (RTC_TPR).....	505
32.3.3	RTC Time Alarm Register (RTC_TAR).....	506
32.3.4	RTC Time Compensation Register (RTC_TCR).....	506
32.3.5	RTC Control Register (RTC_CR).....	508
32.3.6	RTC Status Register (RTC_SR).....	510
32.3.7	RTC Lock Register (RTC_LR).....	511
32.3.8	RTC Interrupt Enable Register (RTC_IER).....	512
32.4	Functional description.....	513

Section number	Title	Page
32.4.1	Power, clocking, and reset.....	513
32.4.2	Time counter.....	514
32.4.3	Compensation.....	514
32.4.4	Time alarm.....	515
32.4.5	Update mode.....	515
32.4.6	Register lock.....	516
32.4.7	Interrupt.....	516

### Chapter 33 Serial Peripheral Interface (SPI)

33.1	Chip-specific SPI information.....	517
33.2	Introduction.....	517
33.2.1	Features.....	518
33.2.2	Modes of operation.....	518
33.2.3	Block diagrams.....	519
33.3	External signal description.....	522
33.3.1	SPSCK — SPI Serial Clock.....	523
33.3.2	MOSI — Master Data Out, Slave Data In.....	523
33.3.3	MISO — Master Data In, Slave Data Out.....	523
33.3.4	SS — Slave Select.....	523
33.4	Memory map/register definition.....	524
33.4.1	SPI Status Register (SPIx_S).....	524
33.4.2	SPI Baud Rate Register (SPIx_BR).....	528
33.4.3	SPI Control Register 2 (SPIx_C2).....	529
33.4.4	SPI Control Register 1 (SPIx_C1).....	531
33.4.5	SPI Match Register low (SPIx_ML).....	532
33.4.6	SPI match register high (SPIx_MH).....	533
33.4.7	SPI Data Register low (SPIx_DL).....	533
33.4.8	SPI data register high (SPIx_DH).....	534
33.4.9	SPI clear interrupt register (SPIx_CI).....	535



Section number	Title	Page
33.4.10	SPI control register 3 (SPIx_C3).....	536
33.5	Functional description.....	538
33.5.1	General.....	538
33.5.2	Master mode.....	538
33.5.3	Slave mode.....	540
33.5.4	SPI FIFO Mode.....	541
33.5.5	SPI Transmission by DMA.....	542
33.5.6	Data Transmission Length.....	544
33.5.7	SPI clock formats.....	545
33.5.8	SPI baud rate generation.....	548
33.5.9	Special features.....	548
33.5.10	Error conditions.....	550
33.5.11	Low-power mode options.....	551
33.5.12	Reset.....	552
33.5.13	Interrupts.....	553
33.6	Initialization/application information.....	555
33.6.1	Initialization sequence.....	555
33.6.2	Pseudo-Code Example.....	556

## Chapter 34 Inter-Integrated Circuit (I2C)

34.1	Chip-specific I2C information.....	561
34.1.1	I2C instantiation information.....	561
34.2	Introduction.....	561
34.2.1	Features.....	562
34.2.2	Modes of operation.....	562
34.2.3	Block diagram.....	562
34.3	I2C signal descriptions.....	563
34.4	Memory map/register definition.....	564
34.4.1	I2C Address Register 1 (I2Cx_A1).....	565

Section number	Title	Page
34.4.2	I2C Frequency Divider register (I2Cx_F).....	565
34.4.3	I2C Control Register 1 (I2Cx_C1).....	566
34.4.4	I2C Status register (I2Cx_S).....	568
34.4.5	I2C Data I/O register (I2Cx_D).....	570
34.4.6	I2C Control Register 2 (I2Cx_C2).....	570
34.4.7	I2C Programmable Input Glitch Filter Register (I2Cx_FLT).....	571
34.4.8	I2C Range Address register (I2Cx_RA).....	573
34.4.9	I2C SMBus Control and Status register (I2Cx_SMB).....	573
34.4.10	I2C Address Register 2 (I2Cx_A2).....	575
34.4.11	I2C SCL Low Timeout Register High (I2Cx_SLTH).....	575
34.4.12	I2C SCL Low Timeout Register Low (I2Cx_SLTL).....	576
34.4.13	I2C Status register 2 (I2Cx_S2).....	576
34.5	Functional description.....	577
34.5.1	I2C protocol.....	577
34.5.2	10-bit address.....	582
34.5.3	Address matching.....	584
34.5.4	System management bus specification.....	585
34.5.5	Resets.....	587
34.5.6	Interrupts.....	587
34.5.7	Programmable input glitch filter.....	590
34.5.8	Address matching wake-up.....	590
34.5.9	DMA support.....	591
34.5.10	Double buffering mode.....	592
34.6	Initialization/application information.....	593

## Chapter 35 Low Power Universal asynchronous receiver/transmitter (LPUART)

35.1	Chip-specific LPUART information.....	597
35.1.1	LPUART0 and LPUART1 overview.....	597
35.2	Introduction.....	597

Section number	Title	Page
35.2.1	Features.....	597
35.2.2	Modes of operation.....	598
35.2.3	Signal Descriptions.....	599
35.2.4	Block diagram.....	599
35.3	Register definition.....	601
35.3.1	LPUART Baud Rate Register (LPUARTx_BAUD).....	602
35.3.2	LPUART Status Register (LPUARTx_STAT).....	604
35.3.3	LPUART Control Register (LPUARTx_CTRL).....	608
35.3.4	LPUART Data Register (LPUARTx_DATA).....	613
35.3.5	LPUART Match Address Register (LPUARTx_MATCH).....	615
35.4	Functional description.....	615
35.4.1	Baud rate generation.....	615
35.4.2	Transmitter functional description.....	616
35.4.3	Receiver functional description.....	618
35.4.4	Additional LPUART functions.....	623
35.4.5	Interrupts and status flags.....	625

## Chapter 36 Universal Asynchronous Receiver/Transmitter(UART)

36.1	Chip-specific UART information.....	627
36.1.1	UART2 Overview.....	627
36.2	Introduction.....	627
36.2.1	Features.....	627
36.2.2	Modes of operation.....	629
36.3	UART signal descriptions.....	630
36.3.1	Detailed signal descriptions.....	630
36.4	Memory map and registers.....	630
36.4.1	UART Baud Rate Registers: High (UARTx_BDH).....	632
36.4.2	UART Baud Rate Registers: Low (UARTx_BDL).....	633
36.4.3	UART Control Register 1 (UARTx_C1).....	633

Section number	Title	Page
36.4.4	UART Control Register 2 (UARTx_C2).....	635
36.4.5	UART Status Register 1 (UARTx_S1).....	637
36.4.6	UART Status Register 2 (UARTx_S2).....	639
36.4.7	UART Control Register 3 (UARTx_C3).....	641
36.4.8	UART Data Register (UARTx_D).....	642
36.4.9	UART Match Address Registers 1 (UARTx_MA1).....	643
36.4.10	UART Match Address Registers 2 (UARTx_MA2).....	644
36.4.11	UART Control Register 4 (UARTx_C4).....	644
36.4.12	UART Control Register 5 (UARTx_C5).....	645
36.4.13	UART 7816 Control Register (UARTx_C7816).....	646
36.4.14	UART 7816 Interrupt Enable Register (UARTx_IE7816).....	647
36.4.15	UART 7816 Interrupt Status Register (UARTx_IS7816).....	649
36.4.16	UART 7816 Wait Parameter Register (UARTx_WP7816).....	651
36.4.17	UART 7816 Wait N Register (UARTx_WN7816).....	651
36.4.18	UART 7816 Wait FD Register (UARTx_WF7816).....	652
36.4.19	UART 7816 Error Threshold Register (UARTx_ET7816).....	652
36.4.20	UART 7816 Transmit Length Register (UARTx_TL7816).....	653
36.4.21	UART 7816 ATR Duration Timer Register A (UARTx_AP7816A_T0).....	653
36.4.22	UART 7816 ATR Duration Timer Register B (UARTx_AP7816B_T0).....	654
36.4.23	UART 7816 Wait Parameter Register A (UARTx_WP7816A_T0).....	655
36.4.24	UART 7816 Wait Parameter Register A (UARTx_WP7816A_T1).....	655
36.4.25	UART 7816 Wait Parameter Register B (UARTx_WP7816B_T0).....	656
36.4.26	UART 7816 Wait Parameter Register B (UARTx_WP7816B_T1).....	656
36.4.27	UART 7816 Wait and Guard Parameter Register (UARTx_WGP7816_T1).....	657
36.4.28	UART 7816 Wait Parameter Register C (UARTx_WP7816C_T1).....	657
36.5	Functional description.....	658
36.5.1	Transmitter.....	658
36.5.2	Receiver.....	662
36.5.3	Baud rate generation.....	674

<b>Section number</b>	<b>Title</b>	<b>Page</b>
36.5.4	Data format (non ISO-7816).....	676
36.5.5	Single-wire operation.....	679
36.5.6	Loop operation.....	679
36.5.7	ISO-7816/smartcard support.....	680
36.6	Reset.....	685
36.7	System level interrupt sources.....	685
36.7.1	RXEDGIF description.....	686
36.8	DMA operation.....	687
36.9	Application information.....	688
36.9.1	ISO-7816 initialization sequence.....	688
36.9.2	Initialization sequence (non ISO-7816).....	689
36.9.3	Overrun (OR) flag implications.....	690
36.9.4	Overrun NACK considerations.....	691
36.9.5	Match address registers.....	692
36.9.6	Clearing 7816 wait timer (WT, BWT, CWT) interrupts.....	692
36.9.7	Legacy and reverse compatibility considerations.....	692

## **Chapter 37 FlexIO**

37.1	Chip-specific FlexIO information.....	695
37.1.1	FlexIO .....	695
37.1.2	Clock options.....	695
37.1.3	Trigger options.....	695
37.2	Introduction.....	696
37.2.1	Overview.....	696
37.2.2	Features.....	696
37.2.3	Block Diagram.....	697
37.2.4	Modes of operation.....	698
37.2.5	FlexIO Signal Descriptions.....	698
37.3	Memory Map and Registers.....	698

Section number	Title	Page
37.3.1	Version ID Register (FLEXIO_VERID).....	700
37.3.2	Parameter Register (FLEXIO_PARAM).....	701
37.3.3	FlexIO Control Register (FLEXIO_CTRL).....	702
37.3.4	Shifter Status Register (FLEXIO_SHIFTSTAT).....	703
37.3.5	Shifter Error Register (FLEXIO_SHIFTEERR).....	704
37.3.6	Timer Status Register (FLEXIO_TIMSTAT).....	704
37.3.7	Shifter Status Interrupt Enable (FLEXIO_SHIFTSIEN).....	705
37.3.8	Shifter Error Interrupt Enable (FLEXIO_SHIFTEIEN).....	706
37.3.9	Timer Interrupt Enable Register (FLEXIO_TIMIEN).....	706
37.3.10	Shifter Status DMA Enable (FLEXIO_SHIFTSDEN).....	707
37.3.11	Shifter Control N Register (FLEXIO_SHIFTCTLn).....	707
37.3.12	Shifter Configuration N Register (FLEXIO_SHIFTCFGn).....	709
37.3.13	Shifter Buffer N Register (FLEXIO_SHIFTBUFn).....	710
37.3.14	Shifter Buffer N Bit Swapped Register (FLEXIO_SHIFTBUFBISn).....	711
37.3.15	Shifter Buffer N Byte Swapped Register (FLEXIO_SHIFTBUFBYSn).....	711
37.3.16	Shifter Buffer N Bit Byte Swapped Register (FLEXIO_SHIFTBUFBBSn).....	712
37.3.17	Timer Control N Register (FLEXIO_TIMCTLn).....	712
37.3.18	Timer Configuration N Register (FLEXIO_TIMCFGn).....	714
37.3.19	Timer Compare N Register (FLEXIO_TIMCMPn).....	716
37.4	Functional description.....	717
37.4.1	Shifter operation.....	717
37.4.2	Timer operation.....	719
37.4.3	Pin operation.....	721
37.5	Application Information.....	722
37.5.1	UART Transmit.....	722
37.5.2	UART Receive.....	723
37.5.3	SPI Master.....	725
37.5.4	SPI Slave.....	727
37.5.5	I2C Master.....	729

Section number	Title	Page
37.5.6	I2S Master.....	731
37.5.7	I2S Slave.....	732

## Chapter 38 General-Purpose Input/Output (GPIO)

38.1	Chip-specific GPIO information.....	735
38.1.1	GPIO instantiation information.....	735
38.1.2	GPIO accessibility in the memory map.....	735
38.2	Introduction.....	736
38.2.1	Features.....	736
38.2.2	Modes of operation.....	736
38.2.3	GPIO signal descriptions.....	737
38.3	Memory map and register definition.....	738
38.3.1	Port Data Output Register (GPIOx_PDOR).....	739
38.3.2	Port Set Output Register (GPIOx_PSOR).....	740
38.3.3	Port Clear Output Register (GPIOx_PCOR).....	740
38.3.4	Port Toggle Output Register (GPIOx_PTOR).....	741
38.3.5	Port Data Input Register (GPIOx_PDIR).....	741
38.3.6	Port Data Direction Register (GPIOx_PDDR).....	742
38.4	FGPIO memory map and register definition.....	742
38.4.1	Port Data Output Register (FGPIOx_PDOR).....	744
38.4.2	Port Set Output Register (FGPIOx_PSOR).....	745
38.4.3	Port Clear Output Register (FGPIOx_PCOR).....	745
38.4.4	Port Toggle Output Register (FGPIOx_PTOR).....	746
38.4.5	Port Data Input Register (FGPIOx_PDIR).....	746
38.4.6	Port Data Direction Register (FGPIOx_PDDR).....	747
38.5	Functional description.....	747
38.5.1	General-purpose input.....	747
38.5.2	General-purpose output.....	747
38.5.3	IOPORT.....	748

Section number	Title	Page
<b>Chapter 39</b>		
<b>Bit Manipulation Engine (BME)</b>		
39.1	Introduction.....	749
39.1.1	Overview.....	750
39.1.2	Features.....	750
39.1.3	Modes of operation.....	751
39.2	Memory map and register definition.....	751
39.3	Functional description.....	751
39.3.1	BME decorated stores.....	752
39.3.2	BME decorated loads.....	759
39.3.3	Additional details on decorated addresses and GPIO accesses.....	765
39.4	Application information.....	766
<b>Chapter 40</b>		
<b>Micro Trace Buffer (MTB)</b>		
40.1	Introduction.....	769
40.1.1	Overview.....	769
40.1.2	Features.....	772
40.1.3	Modes of operation.....	773
40.2	External signal description.....	773
40.3	Memory map and register definition.....	774
40.3.1	MTB_RAM Memory Map.....	774
40.3.2	MTB_DWT Memory Map.....	786
40.3.3	System ROM Memory Map.....	796
<b>Chapter 41</b>		
<b>Cyclic Redundancy Check (CRC)</b>		
41.1	Introduction.....	801
41.1.1	Features.....	801
41.1.2	Block diagram.....	801
41.1.3	Modes of operation.....	802
41.2	Memory map and register descriptions.....	802



Section number	Title	Page
41.2.1	CRC Data register (CRC_DATA).....	803
41.2.2	CRC Polynomial register (CRC_GPOLY).....	804
41.2.3	CRC Control register (CRC_CTRL).....	804
41.3	Functional description.....	805
41.3.1	CRC initialization/reinitialization.....	805
41.3.2	CRC calculations.....	806
41.3.3	Transpose feature.....	807
41.3.4	CRC result complement.....	809

## Chapter 42 Flash Memory Controller (FMC)

42.1	Introduction.....	811
42.1.1	Overview.....	811
42.1.2	Features.....	811
42.2	Modes of operation.....	812
42.3	External signal description.....	812
42.4	Memory map and register descriptions.....	812
42.5	Functional description.....	812

## Chapter 43 Flash Memory Module (FTFA)

43.1	Introduction.....	815
43.1.1	Features.....	815
43.1.2	Block Diagram.....	816
43.1.3	Glossary.....	817
43.2	External Signal Description.....	818
43.3	Memory Map and Registers.....	818
43.3.1	Flash Configuration Field Description.....	818
43.3.2	Program Flash IFR Map.....	819
43.3.3	Register Descriptions.....	819
43.4	Functional Description.....	829

<b>Section number</b>	<b>Title</b>	<b>Page</b>
43.4.1	Flash Protection.....	829
43.4.2	Interrupts.....	829
43.4.3	Flash Operation in Low-Power Modes.....	830
43.4.4	Functional Modes of Operation.....	831
43.4.5	Flash Reads and Ignored Writes.....	831
43.4.6	Read While Write (RWW).....	831
43.4.7	Flash Program and Erase.....	831
43.4.8	Flash Command Operations.....	832
43.4.9	Margin Read Commands.....	836
43.4.10	Flash Command Description.....	837
43.4.11	Security.....	851
43.4.12	Reset Sequence.....	853

# Chapter 1

## About This Document

### 1.1 Overview

#### 1.1.1 Purpose

This document describes the features, architecture, and programming model of the NXP KL13 microcontroller.

#### 1.1.2 Audience

A reference manual is primarily for system architects and software application developers who are using or considering using a NXP product 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.

# Chapter 2

## Introduction

### 2.1 Overview

Information found here provides an overview of the Kinetis L series of ARM® Cortex®-M0+ MCUs and KL13 product family. It also presents high-level descriptions of the modules available on the devices covered by this document.

#### 2.1.1 Sub-family introduction

The device is highly-integrated, market leading ultra low-power 32-bit microcontroller based on the enhanced Cortex-M0+ (CM0+) core platform. The features of the family derivatives are as follows.

- Core platform clock up to 48 MHz, bus clock up to 24 MHz
- Memory option is up to 64 KB flash and 8 KB RAM
- Wide operating voltage ranges from 1.71–3.6 V with fully functional flash program/erase/read operations
- Multiple package options from 32-pin to 80-pin
- Ambient operating temperature ranges from –40 °C to 105 °C.

The family acts as an ultra low-power, cost-effective microcontroller to provide developers an appropriate entry-level 32-bit solution. The family is the next-generation MCU solution for low-cost, low-power, high-performance devices applications. It's valuable for cost-sensitive, portable applications requiring long battery life-time.

## 2.2 Module functional categories

The modules on this device are grouped into functional categories. Information found here describes the modules assigned to each category in more detail.

**Table 2-1. Module functional categories**

Module category	Description
ARM Cortex-M0+ core	<ul style="list-style-type: none"> <li>32-bit MCU core from ARM's Cortex-M class, 1.77 CoreMark®/MHz from single-cycle access memories, 48 MHz CPU frequency</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 run, wait, stop, and power-down modes</li> </ul> </li> <li>Miscellaneous control module</li> <li>Low-leakage wakeup unit</li> <li>Peripheral bridge</li> <li>Direct memory access (DMA) controller with multiplexer to increase available DMA requests</li> <li>COP watchdog</li> </ul>
Memories	<ul style="list-style-type: none"> <li>Internal memories include:               <ul style="list-style-type: none"> <li>Up to 64 KB flash memory</li> <li>Up to 8 KB SRAM</li> <li>8 KB boot ROM</li> </ul> </li> </ul>
Clocks	<ul style="list-style-type: none"> <li>Multiple clock generation options available from internally- and externally-generated clocks               <ul style="list-style-type: none"> <li>MCG-Lite with 48MIRC and 8M/2M IRC for systems and CPU clock sources.</li> <li>Low power 1 kHz RC oscillator for RTC and COP watchdog</li> </ul> </li> <li>System oscillator to provide clock source for the MCU</li> </ul>
Security	<ul style="list-style-type: none"> <li>COP watchdog timer (COP)</li> <li>Cyclic Redundancy Check (CRC)</li> </ul>
Analog	<ul style="list-style-type: none"> <li>16-bit analog-to-digital converters with DMA supported and four muxed differential pairs</li> <li>Comparator (CMP) with internal 6-bit digital-to-analog converter (DAC)</li> <li>12-bit DAC with DMA support and two 16-bit data buffer</li> <li>High accuracy 1.2 V voltage reference to provide a stable reference for ADC</li> </ul>
Timers	<ul style="list-style-type: none"> <li>One 6-channel TPM</li> <li>Two 2-channel TPMs</li> <li>2-channel periodic interrupt timer</li> <li>Real time clock</li> <li>Low-power timer</li> <li>System tick timer</li> </ul>
Communications	<ul style="list-style-type: none"> <li>Two 16-bit serial peripheral interface</li> <li>Two inter-integrated circuit (I<sup>2</sup>C) modules</li> <li>Two low power UART modules and one UART module</li> <li>One FlexIO</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-M0+ core modules

The following core modules are available on this device.

**Table 2-2. Core modules**

Module	Description
ARM Cortex-M0+	The ARM Cortex-M0+ is the newest member of the Cortex M Series of processors targeting microcontroller applications focused on very cost sensitive, deterministic, interrupt driven environments. The Cortex M0+ processor is based on the ARMv6 Architecture and Thumb@-2 ISA and is 100% instruction set compatible with its predecessor, the Cortex-M0 core, and upward compatible to Cortex-M3 and M4 cores.
NVIC	The ARMv6-M exception model and nested-vector interrupt controller (NVIC) implement a relocatable vector table supporting many external interrupts, a single non-maskable interrupt (NMI), and priority levels.  The NVIC replaces shadow registers with equivalent system and simplified programmability. The NVIC contains the address of the function to execute for a particular handler. The address is fetched via the instruction port allowing parallel register stacking and look-up. The first sixteen entries are allocated to ARM internal sources with the others mapping to MCU-defined interrupts.
AWIC	The primary function of the Asynchronous Wake-up Interrupt Controller (AWIC) is to detect asynchronous wake-up events in stop modes and signal to clock control logic to resume system clocking. After clock restart, the NVIC observes the pending interrupt and performs the normal interrupt or event processing.
Single-cycle I/O Port	For high-speed, single-cycle access to peripherals, the Cortex-M0+ processor implements a dedicated single-cycle I/O port.
Debug interfaces	Most of this device's debug is based on the ARM CoreSight™ architecture. One debug interface is supported: <ul style="list-style-type: none"> <li>Serial Wire Debug (SWD)</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 (SMC)	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. Multiple modes are supported that allow the user to optimize power consumption for the level of functionality needed. Includes power-on-reset (POR) and integrated low voltage detect (LVD) with reset (brownout) capability and selectable LVD trip points.

*Table continues on the next page...*

**Table 2-3. System modules (continued)**

Module	Description
Miscellaneous control module (MCM)	The MCM includes integration logic and details.
Crossbar switch lite (AXBS-Lite)	The AXBS 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.
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.
Peripheral bridge (AIPS-Lite)	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 4 for the DMA controller.
Direct memory access (DMA) controller	The DMA controller provides programmable channels with transfer control descriptors for data movement via dual-address transfers for 8-, 16- and 32-bit data values.
Computer operating properly 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, 8/2 MHz internal oscillator or external crystal 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	Program flash memory — up to 64 KB of the non-volatile flash memory that can execute program code.
Flash memory controller	Manages the interface between the device and the on-chip flash memory.
SRAM	Up to 8 KB internal system RAM.
ROM	8 KB ROM.

## 2.2.4 Clocks

The following clock modules are available on this device.

**Table 2-5. Clock modules**

Module	Description
Multipurpose Clock Generator Lite (MCG-Lite)	MCG Lite module containing a 48 MHz and an 8 or 2 MHz internal reference clock source.
System oscillator (OSC)	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
Watchdog timer (WDOG)	Watchdog timer keeps a watch on the system functioning and resets it in case of its failure.
Cyclic Redundancy Check (CRC)	CRC generates 16/32-bit CRC code for error detection.

## 2.2.6 Analog modules

The following analog modules are available on this device:

**Table 2-7. Analog modules**

Module	Description
Analog-to-digital converters (ADC)	16-bit successive-approximation ADC module.
Analog comparators	One comparator that compares two analog input voltages across the full range of the supply voltage and can trigger an ADC acquisition, TPM update, or CPU interrupt.
6-bit digital-to-analog converters (DAC)	64-tap resistor ladder network which provides a selectable voltage reference for comparator.
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.
12-bit digital-to-analog converters (DAC)	Low-power general-purpose DAC, whose output can be placed on an external pin or set as one of the inputs to the analog comparator or ADC.

## 2.2.7 Timer modules

The following timer modules are available on this device:

**Table 2-8. Timer modules**

Module	Description
Timer/PWM module (TPM)	<ul style="list-style-type: none"> <li>Selectable TPM clock mode</li> <li>Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128</li> <li>16-bit free-running counter or modulo counter with counting be up or up-down</li> </ul>

*Table continues on the next page...*

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

Module	Description
	<ul style="list-style-type: none"> <li>configurable channels for input capture, output compare, edge-aligned PWM mode, or center-aligned PWM mode</li> <li>Support the generation of an interrupt and/or DMA request per channel</li> <li>Support the generation of an interrupt and/or DMA request when the counter overflows</li> <li>Support selectable trigger input to optionally reset or cause the counter to start incrementing.</li> <li>Support the generation of hardware triggers when the counter overflows and per channel</li> </ul>
Periodic interrupt timers (PIT)	<ul style="list-style-type: none"> <li>One general purpose interrupt timer</li> <li>Interrupt timers for triggering ADC conversions</li> <li>32-bit counter resolution</li> <li>Clocked by bus clock frequency</li> <li>DMA support</li> </ul>
Low power timer (LPTMR)	<ul style="list-style-type: none"> <li>16-bit time counter or pulse counter with compare</li> <li>Configurable clock source for prescaler/glitch filter</li> <li>Configurable input source for pulse counter</li> </ul>
Real Time Clock (RTC)	<ul style="list-style-type: none"> <li>32-bit seconds counter with roll-over protection and 32-bit alarm</li> <li>Software selectable clock sources for input to prescaler with programmable 16-bit prescaler <ul style="list-style-type: none"> <li>XOSC 32.678 kHz nominal</li> <li>LPO (~1 kHz)</li> <li>External RTC_CLKIN</li> </ul> </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) and (LPUART)	Two low power UART modules that retains functional in stop modes. One UART module does not work in stop mode.
FlexIO	The FlexIO module is capable of supporting a wide range of protocols including, but not limited to--UART, I2C, SPI, I2S, Camera IF, LCD RGB, PWM / Waveform generation. The module can remain functional in VLPS mode provided the clock it is using remains enabled.

## 2.2.9 Human-machine interfaces

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

Table 2-10. HMI modules

Module	Description
General purpose input/output (GPIO)	Some general purpose input or output (GPIO) pins are capable of interrupt and DMA request generation.

## 2.3 Module to module interconnects

### 2.3.1 Interconnection overview

The following table lists the module to module interconnections for this device.

Table 2-11. Module-to-module interconnects

Peripheral	Signal	—	to Peripheral	Use Case	Control	Comment
TPM1	CH0F, CH1F	to	ADC (Trigger)	ADC Triggering (A AND B)	SIM_SOPT7[ADC0ALTTRGEN] = 0	Ch0 is A, and Ch1 is B, selecting this ADC trigger is for supporting A and B triggering. In Stop and VLPS modes, the second trigger must be set to >10 $\mu$ s after the first trigger
LPTMR	Hardware trigger	to	ADC (Trigger)	ADC Triggering (A or B)	SIM_SOPT7[ADC0TRGSEL] and SIM_SOPT7[ADC0PRETRGSEL] to select A or B	—
TPMx	TOF	to	ADC (Trigger)	ADC Triggering (A or B)	SIM_SOPT7[ADC0TRGSEL], SIM_SOPT7[ADC0PRETRGSEL] to select A or B	—
PIT CHx	TIF0, TIF1	to	ADC (Trigger)	ADC Triggering (A or B)	SIM_SOPT7[ADC0TRGSEL], SIM_SOPT7[ADC0PRETRGSEL] to select A or B	—
RTC	ALARM or SECONDS	to	ADC (Trigger)	ADC Triggering (A or B)	SIM_SOPT7[ADC0TRGSEL], SIM_SOPT7[ADC0PRETRGSEL] to select A or B	—
EXTRG_IN	EXTRG_IN	to	ADC (Trigger)	ADC Triggering (A or B)	SIM_SOPT7[ADC0TRGSEL], SIM_SOPT7[ADC0PRETRGSEL] to select A or B	—
CMP0	CMP0_OUT	to	ADC (Trigger)	ADC Triggering (A or B)	SIM_SOPT7[ADC0TRGSEL], SIM_SOPT7[ADC0PRETRGSEL] to select A or B	—
CMP0	CMP0_OUT	to	LPTMR_ALT 0	Count CMP events	LPTMR_CSR[TPS]	—
CMP0	CMP0_OUT	to	TPM1 CH0	Input capture	SIM_SOPT4[TPM1CH0SRC]	—

Table continues on the next page...

Table 2-11. Module-to-module interconnects (continued)

Peripheral	Signal	—	to Peripheral	Use Case	Control	Comment
CMP0	CMP0_OUT	to	TPM2 CH0	Input capture	SIM_SOPT4[TPM2CH0SRC]	—
CMP0	CMP0_OUT	to	LPUART0_RX	IR interface	SIM_SOPT5[LPUART0RXSRC]	Uses for IR interface
CMP0	CMP0_OUT	to	LPUART1_RX	IR Interface	SIM_SOPT5[LPUART1RXSRC]	Uses for IR interface
LPTMR	Hardware trigger	to	CMPx	Low power triggering of the comparator	CMP_CR1[TRIGM]	—
LPTMR	Hardware trigger	to	TPMx	TPM Trigger input	TPMx_CONF[TRGSEL] (4-bit field)	—
TPMx	TOF	to	TPMx	TPM Trigger input	TPMx_CONF[TRGSEL] (4-bit field)	—
TPM1	Timebase	to	TPMx	TPM Global timebase input	TPMx_CONF[GTBEEN]	—
PIT CHx	TIF0, TIF1	to	TPMx	TPM Trigger input	TPMx_CONF[TRGSEL] (4-bit field)	If PIT is triggering the TPM, the TPM clock must be faster than Bus clock.
RTC	ALARM or SECONDS	to	TPMx	TPM Trigger input	TPMx_CONF[TRGSEL] (4-bit field)	—
EXTRG_IN	EXTRG_IN	to	TPMx	TPM Trigger input	TPMx_CONF[TRGSEL] (4-bit field)	—
CMP0	CMP0_OUT	to	TPMx	TPM Trigger input	TPMx_CONF[TRGSEL] (4-bit field)	—
LPUART0	LPUART0_TX	to	Modulated by TPM1 CH0	LPUART modulation	SIM_SOPT5[LPUART0TXSRC]	Uses for IR interface
LPUART0	LPUART0_TX	to	Modulated by TPM2 CH0	LPUART modulation	SIM_SOPT5[LPUART0TXSRC]	Uses for IR interface
LPUART1	LPUART1_TX	to	Modulated by TPM1 CH0	LPUART modulation	SIM_SOPT5[LPUART1TXSRC]	Uses for IR interface
LPUART1	LPUART1_TX	to	Modulated by TPM2 CH0	LPUART modulation	SIM_SOPT5[LPUART1TXSRC]	Uses for IR interface
PIT	TIF0	to	DAC	Advance DAC FIFO	DAC HWTRG Select	—
PIT	TIF0	to	DMA CH0	DMA HW Trigger	DMA MUX register option	—
PIT	TIF1	to	DMA CH1	DMA HW Trigger	DMA MUX register option	—

Table 2-12. Module-to-FlexIO interconnects

Peripheral	Signal	—	to Peripheral	Use Case	Control	Comment
LPTMR	Hardware trigger	to	FlexIO	Trigger input	FlexIO_TIMCTLn[TRGSEL] (4-bit field)	—
TPMx	TOF	to	FlexIO	Trigger input	FlexIO_TIMCTLn[TRGSEL] (4-bit field)	—
PIT CHx	TIF0, TIF1	to	FlexIO	Trigger input	FlexIO_TIMCTLn[TRGSEL] (4-bit field)	If PIT is triggering the FlexIO, the FlexIO clock must be faster than Bus clock.
RTC	ALARM or SECONDS	to	FlexIO	Trigger input	FlexIO_TIMCTLn[TRGSEL] (4-bit field)	—
EXTRG_IN	EXTRG_IN	to	FlexIO	Trigger input	FlexIO_TIMCTLn[TRGSEL] (4-bit field)	—
CMP0	CMP0_OUT	to	FlexIO	Trigger input	FlexIO_TIMCTLn[TRGSEL] (4-bit field)	—

## 2.3.2 Analog reference options

Several analog blocks have selectable reference voltages as shown in the below table . These options allow analog peripherals to share or have separate analog references. Care should be taken when selecting analog references to avoid cross talk noise.

Table 2-13. Analog reference options

Module	Reference option	Comment/ Reference selection
16-bit SAR ADC	1 - VREFH or 1.2V VREF_OUT 2 - VDDA 3 - Reserved	Selected by ADCx_SC2[REFSEL]
12-bit DAC	1 - VREFH or 1.2V VREF_OUT 2 - VDDA <sup>1</sup>	Selected by DACx_C0[DACRFS] bit
CMP with 6-bit DAC	Vin1 - VREFH or 1.2V VREF_OUT Vin2 - VDD <sup>1</sup>	Selected by CMPx_DACCR[VRSEL]

1. Use this option for the best ADC operation.

### NOTE

VREFH pin can be used as filter capacitor pin for high precision 1.2V VREF\_OUT. When 1.2V VREF is enabled, VREFH is 1.2V VREF\_OUT. VREF is only available in 48pin or above package.



# Chapter 3

## Core Overview

### 3.1 ARM Cortex-M0+ core introduction

The enhanced ARM Cortex M0+ is the member of the Cortex-M Series of processors targeting microcontroller cores focused on very cost sensitive, low power applications. It has a single 32-bit AMBA AHB-Lite interface and includes an NVIC component. It also has hardware debug functionality including support for simple program trace capability. The processor supports the ARMv6-M instruction set (Thumb) architecture including all but three 16-bit Thumb opcodes (52 total) plus seven 32-bit instructions. It is upward compatible with other Cortex-M profile processors.

#### 3.1.1 Buses, interconnects, and interfaces

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

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

#### 3.1.2 System tick timer

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

#### 3.1.3 Debug facilities

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

### 3.1.4 Core privilege levels

The core on this device is implemented with both privileged and unprivileged levels. The ARM documentation uses different terms than this document to distinguish between privilege levels.

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

## 3.2 Nested vectored interrupt controller (NVIC)

### 3.2.1 Interrupt priority levels

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

### 3.2.2 Non-maskable interrupt

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

### 3.2.3 Interrupt channel assignments

The interrupt vector assignments are defined in the following table.

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

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



**NOTE**

The NVIC wake-up sources in the following table support only down to VLPS.

**Table 3-2. Interrupt vector assignments**

Address	Vector	IRQ <sup>1</sup>	NVIC IPR register number <sup>2</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	—	—	—	—
0x0000_0014	5	—	—	—	—
0x0000_0018	6	—	—	—	—
0x0000_001C	7	—	—	—	—
0x0000_0020	8	—	—	—	—
0x0000_0024	9	—	—	—	—
0x0000_0028	10	—	—	—	—
0x0000_002C	11	—	—	ARM core	Supervisor call (SVCall)
0x0000_0030	12	—	—	—	—
0x0000_0034	13	—	—	—	—
0x0000_0038	14	—	—	ARM core	Pendable request for system service (PendableSrvReq)
0x0000_003C	15	—	—	ARM core	System tick timer (SysTick)
<b>Non-Core Vectors</b>					
0x0000_0040	16	0	0	DMA	DMA channel 0 transfer complete and error
0x0000_0044	17	1	0	DMA	DMA channel 1 transfer complete and error
0x0000_0048	18	2	0	DMA	DMA channel 2 transfer complete and error
0x0000_004C	19	3	0	DMA	DMA channel 3 transfer complete and error
0x0000_0050	20	4	1	—	—
0x0000_0054	21	5	1	FTFA	Command complete and read collision
0x0000_0058	22	6	1	PMC	Low-voltage detect, low-voltage warning
0x0000_005C	23	7	1	LLWU	Low Leakage Wakeup
0x0000_0060	24	8	2	I <sup>2</sup> C0	Status and Timeout and wakeup flags
0x0000_0064	25	9	2	I <sup>2</sup> C1	Status and Timeout and wakeup flags
0x0000_0068	26	10	2	SPI0	Single interrupt vector for all sources
0x0000_006C	27	11	2	SPI1	Single interrupt vector for all sources
0x0000_0070	28	12	3	LPUART0	Status and error
0x0000_0074	29	13	3	LPUART1	Status and error
0x0000_0078	30	14	3	UART2 or FlexIO	Status and error

Table continues on the next page...

**Table 3-2. Interrupt vector assignments (continued)**

Address	Vector	IRQ <sup>1</sup>	NVIC IPR register number <sup>2</sup>	Source module	Source description
0x0000_007C	31	15	3	ADC0	Conversion complete
0x0000_0080	32	16	4	CMP0	Rising or falling edge of comparator output
0x0000_0084	33	17	4	TPM0	Overflow or channel interrupt
0x0000_0088	34	18	4	TPM1	Overflow or channel interrupt
0x0000_008C	35	19	4	TPM2	Overflow or channel interrupt
0x0000_0090	36	20	5	RTC	Alarm interrupt
0x0000_0094	37	21	5	RTC	Seconds interrupt
0x0000_0098	38	22	5	PIT	Single interrupt vector for all channels
0x0000_009C	39	23	5	—	—
0x0000_00A0	40	24	6	—	—
0x0000_00A4	41	25	6	DAC0	—
0x0000_00A8	42	26	6	—	—
0x0000_00AC	43	27	6	—	—
0x0000_00B0	44	28	7	LPTMR0	LP Timer compare match
0x0000_00B4	45	29	7	—	—
0x0000_00B8	46	30	7	Port control module	Pin detect (Port A)
0x0000_00BC	47	31	7	Port control module	Pin detect (Single interrupt vector for Port B, Port C, Port D, and Port E)

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

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

Suppose you need to configure the SPI0 interrupt. The following table is an excerpt of the SPI0 row from [Interrupt priority levels](#).

**Table 3-3. Interrupt vector assignments**

Address	Vector	IRQ <sup>1</sup>	NVIC IPR register number <sup>2</sup>	Source module	Source description
0x0000_0068	26	10	2	SPI0	Single interrupt vector for all sources

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

- The NVIC registers you would use to configure the interrupt are:

- NVICIPR2
- To determine the particular IRQ's field location within these particular registers:
  - NVICIPR2 field starting location =  $8 * (\text{IRQ mod } 4) + 6 = 22$

Since the NVICIPR fields are 2-bit wide (4 priority levels), the NVICIPR2 field range is 22–23.

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

- NVICIPR2[23:22]

### 3.3 AWIC introduction

The primary function of the AWIC block 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.

#### 3.3.1 Wake-up sources

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

**Table 3-4. AWIC stop wake-up sources**

Wake-up source	Description
Available system resets	RESET pin with filter mode disabled or enabled when LPO is its clock source, COP when its clock source is enabled. COP can also work when its clock source is enabled during STOP mode.
Low-voltage detect	Power management controller—functional in Stop mode
Low-voltage warning	Power management controller—functional in Stop mode
Pin interrupts	Port control module—any enabled pin interrupt is capable of waking the system
ADC	The ADC is functional when using internal clock source or external crystal clock
CMP0	Interrupt in normal or trigger mode
I <sup>2</sup> C	Address match wakeup
LPUART0 , LPUART1	Any enabled interrupt can be a source as long as the module remains clocked
UART2	Active edge on RXD
RTC	Alarm or seconds interrupt
NMI	NMI pin
TPMx	Any enabled interrupt can be a source as long as the module remains clocked
LPTMR	Any enabled interrupt can be a source as long as the module remains clocked
SPIx	Slave mode interrupt
FlexIO	Any enabled interrupt can be a source as long as the module remains clocked



# Chapter 4

## Memory Map

### 4.1 Introduction

This device contains various memories and memory-mapped peripherals which are located in a 4 GB memory space.

This chapter describes the memory and peripheral locations within that memory space.

### 4.2 Flash memory

The devices covered in this document contain 1 program flash block consisting of 1 KB sectors.

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

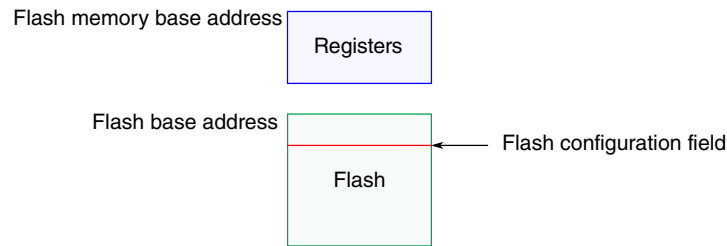
**Table 4-1. KL13 Flash Memory Size**

Part number	Program flash (KB)	Block 0 (P-Flash) address range
MKL13Z32VLK4, MKL13Z32VLH4	32KB	0x0000_0000 – 0x0000_7FFF
MKL13Z64VLK4, MKL13Z64VLH4	64KB	0x0000_0000 – 0x0000_FFFF

#### 4.2.1 Flash memory map

The flash memory and the flash registers are located at different base addresses as shown in the figure found here.

The base address for each is specified in [System memory map](#).



**Figure 4-1. Flash memory map**

The on-chip flash memory 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](#) for details of supported ranges.

Access 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.

## 4.2.2 Flash security

For information on how flash security is implemented on this device, see [Security](#).

## 4.2.3 Flash modes

The flash memory chapter defines two modes of operation: NVM normal and NVM special modes. On this device, the flash memory only operates in NVM normal mode. All references to NVM special mode must be ignored.

## 4.2.4 Erase all flash contents

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

## 4.2.5 FTFA\_FOPT register

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

## 4.3 SRAM

### 4.3.1 SRAM sizes

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

**Table 4-2. KL13 SRAM memory size**

Device	SRAM(KB)
MKL13Z32VLH4(R)	4KB
MKL13Z64VLH4(R)	8KB
MKL13Z32VLK4(R)	4KB
MKL13Z64VLK4(R)	8KB

### 4.3.2 SRAM ranges

The on-chip SRAM is split into two ranges, 1/4 is allocated SRAM\_L and 3/4 is allocated to SRAM\_U.

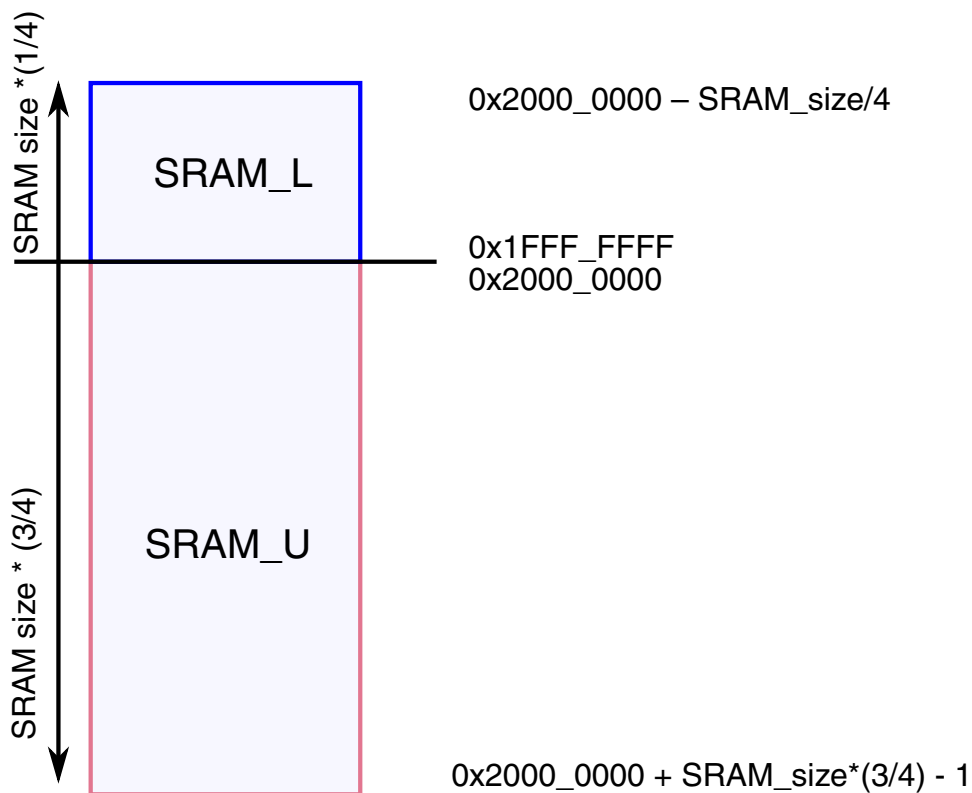
The on-chip RAM is implemented such that the SRAM\_L and SRAM\_U ranges form a contiguous block in the memory map. As such:

- SRAM\_L is anchored to 0x1FFF\_FFFF and occupies the space before this ending address.
- SRAM\_U is anchored to 0x2000\_0000 and occupies the space after this beginning address.

Valid address ranges for SRAM\_L and SRAM\_U are then defined as:

- SRAM\_L = [0x2000\_0000–(SRAM\_size/4)] to 0x1FFF\_FFFF
- SRAM\_U = 0x2000\_0000 to [0x2000\_0000+(SRAM\_size\*(3/4))-1]

This is illustrated in the following figure.



**Figure 4-2. SRAM blocks memory map**

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

- SRAM\_L:  $0x1FFF\_F000 - 0x1FFF\_FFFF$
- SRAM\_U:  $0x2000\_0000 - 0x2000\_2FFF$

### 4.3.3 SRAM retention in low power modes

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

## 4.4 System Register file

This device includes a 32-byte register file that is powered in all power modes.

Also, it retains contents during low-voltage detect (LVD) events and is only reset during a power-on reset.



## 4.5 System memory map

The table found here shows the high-level device memory map.

**Table 4-3. 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 192 bytes)	All masters
0x0800_0000–0x1BFF_FFFF	Reserved	—
0x1C00_0000 – 0x1C00_1FFF	Boot ROM	Cortex-M0+ core
0x1C00_2000 – 0x1FFF_F7FF	Reserved	—
0x1FFF_F800–0x1FFF_FFFF <sup>2</sup>	SRAM_L: Lower SRAM	All masters
0x2000_0000–0x2000_17FF <sup>2</sup>	SRAM_U: Upper SRAM	All masters
0x2000_1800–0x3FFF_FFFF	Reserved	–
0x4000_0000–0x4007_FFFF	AIPS Peripherals	Cortex-M0+ core & DMA
0x4008_0000–0x400F_EFFF	Reserved	–
0x400F_F000–0x400F_FFFF	General-purpose input/output (GPIO)	Cortex-M0+ core & DMA
0x4010_0000–0x43FF_FFFF	Reserved	–
0x4400_0000–0x5FFF_FFFF	Bit Manipulation Engine (BME) access to AIPS Peripherals for slots 0-127 <sup>3</sup>	Cortex-M0+ core
0x6000_0000–0xDFFF_FFFF	Reserved	–
0xE000_0000–0xE00F_FFFF	Private Peripherals	Cortex-M0+ core
0xE010_0000–0xEFFF_FFFF	Reserved	–
0xF000_0000–0xF000_0FFF	Micro Trace Buffer (MTB) registers	Cortex-M0+ core
0xF000_1000–0xF000_1FFF	MTB Data Watchpoint and Trace (MTBDWT) registers	Cortex-M0+ core
0xF000_2000–0xF000_2FFF	ROM table	Cortex-M0+ core
0xF000_3000–0xF000_3FFF	Miscellaneous Control Module (MCM)	Cortex-M0+ core
0xF000_4000–0xF7FF_FFFF	Reserved	–
0xF800_0000–0xFFFF_FFFF	IOPORT: GPIO (single cycle)	Cortex-M0+ core

1. The program flash always begins at 0x0000\_0000 but the end of implemented flash varies depending on the amount of flash implemented for a particular device. See [Flash memory](#) for details.
2. This range varies depending on SRAM sizes. See [SRAM sizes](#) for details.
3. Includes BME operations to GPIO at slot 15 (based at 0x4000\_F000).

## 4.6 Bit Manipulation Engine

The Bit Manipulation Engine (BME) provides hardware support for atomic read-modify-write memory operations to the peripheral address space.

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

### 4.7 Peripheral bridge (AIPS-Lite) memory map

The peripheral memory map is accessible via one slave port on the crossbar in the 0x4000\_0000–0x400F\_FFFF region. The device implements one peripheral bridge that defines a 1024 KB address space.

The three regions associated with this space are:

- A 128 KB region, partitioned as 32 spaces, each 4 KB in size and reserved for on-platform peripheral devices. The AIPS controller generates unique module enables for all 32 spaces.
- A 384 KB region, partitioned as 96 spaces, each 4 KB in size and reserved for off-platform modules. The AIPS controller generates unique module enables for all 96 spaces.
- The last slot is a 4 KB region beginning at 0x400F\_F000 for accessing the GPIO module. The GPIO slot (slot 128) is an alias of slot 15. This block is also directly interfaced to the core and provides direct access without incurring wait states associated with accesses via the AIPS controller.

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

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

#### 4.7.1 Read-after-write sequence and required serialization of memory operations

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

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

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

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

## 4.7.2 Peripheral bridge (AIPS-Lite) memory map

Table 4-4. Peripheral bridge 0 slot assignments

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

Table continues on the next page...

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

System 32-bit base address	Slot number	Module
0x4001_E000	30	—
0x4001_F000	31	—
0x4002_0000	32	Flash memory
0x4002_1000	33	DMA channel multiplexer 0
0x4002_2000	34	—
0x4002_3000	35	—
0x4002_4000	36	—
0x4002_5000	37	—
0x4002_6000	38	—
0x4002_7000	39	—
0x4002_8000	40	—
0x4002_9000	41	—
0x4002_A000	42	—
0x4002_B000	43	—
0x4002_C000	44	—
0x4002_D000	45	—
0x4002_E000	46	—
0x4002_F000	47	—
0x4003_0000	48	—
0x4003_1000	49	—
0x4003_2000	50	CRC32
0x4003_3000	51	—
0x4003_4000	52	—
0x4003_5000	53	—
0x4003_6000	54	—
0x4003_7000	55	Periodic interrupt timers (PIT)
0x4003_8000	56	Timer/PWM (LPTPM) 0
0x4003_9000	57	Timer/PWM (LPTPM) 1
0x4003_A000	58	Timer/PWM (LPTPM) 2
0x4003_B000	59	Analog-to-digital converter 0(ADC0)
0x4003_C000	60	—
0x4003_D000	61	Real Time Clock (RTC)
0x4003_E000	62	—
0x4003_F000	63	DAC0
0x4004_0000	64	Low-power timer (LPTMR)
0x4004_1000	65	System register file
0x4004_2000	66	—
0x4004_3000	67	—
0x4004_4000	68	—

*Table continues on the next page...*

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

System 32-bit base address	Slot number	Module
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	—
0x4005_1000	81	—
0x4005_2000	82	—
0x4005_3000	83	—
0x4005_4000	84	LPUART0
0x4005_5000	85	LPUART1
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	FlexIO
0x4006_0000	96	—
0x4006_1000	97	—
0x4006_2000	98	—
0x4006_3000	99	—
0x4006_4000	100	Multi-purpose clock Generator Lite (MCG_Lite)
0x4006_5000	101	System oscillator (OSC)
0x4006_6000	102	I <sup>2</sup> C0
0x4006_7000	103	I <sup>2</sup> C1
0x4006_8000	104	—
0x4006_9000	105	—
0x4006_A000	106	—
0x4006_B000	107	—

*Table continues on the next page...*

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

System 32-bit base address	Slot number	Module
0x4006_C000	108	UART2
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	SPI0
0x4007_7000	119	SPI1
0x4007_8000	120	—
0x4007_9000	121	—
0x4007_A000	122	—
0x4007_B000	123	—
0x4007_C000	124	Low-leakage wakeup unit (LLWU)
0x4007_D000	125	Power management controller (PMC)
0x4007_E000	126	System mode controller (SMC)
0x4007_F000	127	Reset control module (RCM)
0x400F_F000	128	GPIO controller

# Chapter 5

## Clock Distribution

### 5.1 Introduction

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

The Cortex M0+ resides within a synchronous core platform, where the processor and bus masters, flash memory, and peripheral clocks can be configured independently. The clock distribution figure shows how clocks from the lite version of Multi Clock Generation (MCG-Lite) and OSC module are distributed to the microcontroller's other function units. Some modules in the microcontroller have selectable clock input.

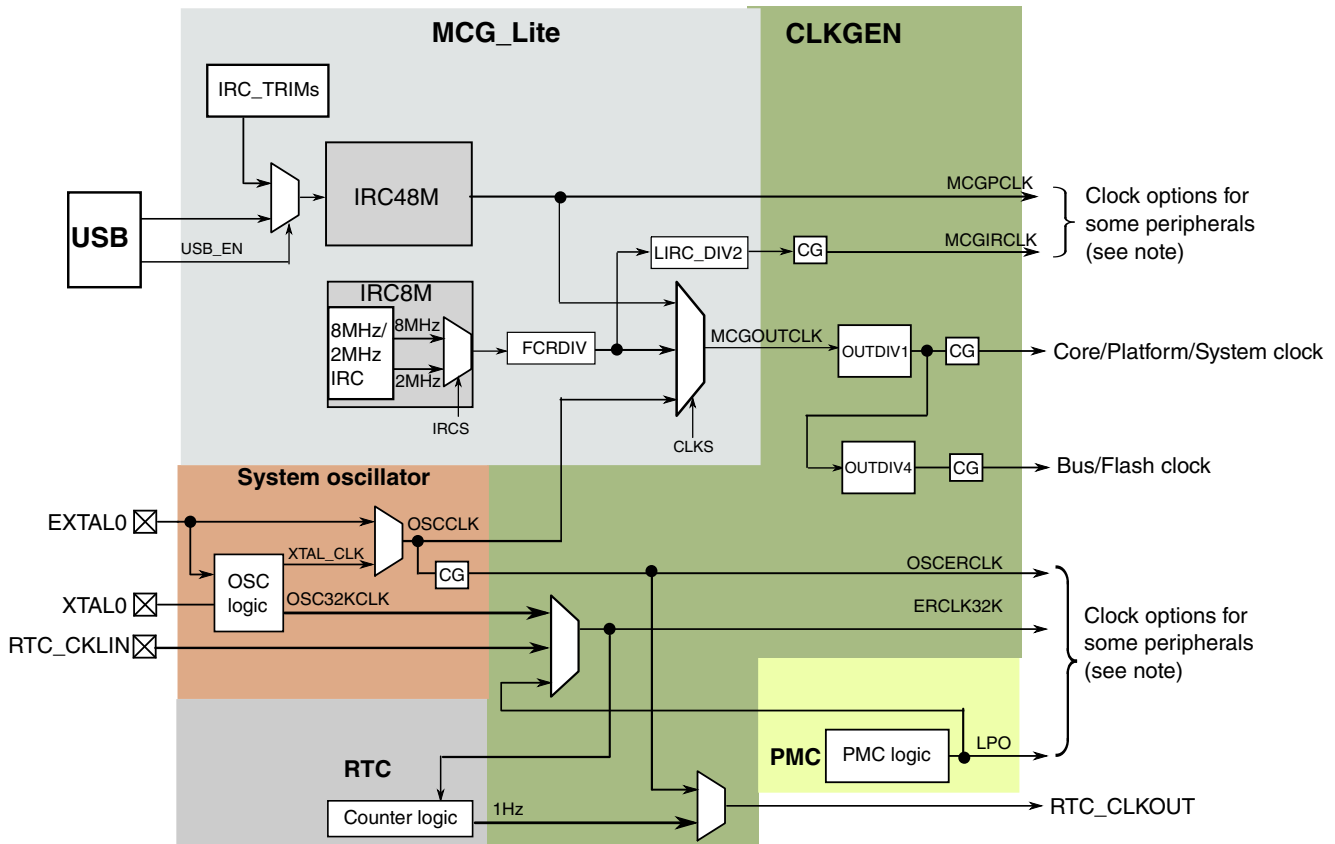
### 5.2 Programming model

The selection and multiplexing of system clock sources is controlled and programmed via the Clock Generation Module. The setting of clock dividers and module clock gating for the system are programmed via the SIM module. Refer to the [MCG\\_Lite](#) and [SIM](#) sections for detailed register and bit descriptions.

### 5.3 High-level device clocking diagram

The following [system oscillator](#), [MCG\\_Lite](#), and [SIM](#) module registers control the multiplexers, dividers, and clock gates shown in the following figure:

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



CG — Clock gate

**Note1:** See subsequent sections for details on where these clocks are used.

**Note2:** 48Mhz clock (IRC48M) control register is defined in either USB or MCG\_Lite. In case USB is not available, IRC48M will be controlled by IRC\_TRIMs in MCG\_Lite module

**Note3:** FCRDIV support divider ratio 1,2,4,8,16, 32, 64, 128. LIRC\_DIV2 provides the further divide down for MCGIRCLK.

**Note4:** OSC32KCLK is only available when external crystal is in 30KHz - 40KHz range.

**Figure 5-1. Clocking diagram**

## 5.4 Clock definitions

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

Clock name	Description
Core clock	MCGOUTCLK divided by OUTDIV1 Clocks the ARM Cortex-M0+ core.
Platform clock	MCGOUTCLK divided by OUTDIV1 Clocks the crossbar switch and NVIC.
System clock	MCGOUTCLK divided by OUTDIV1 Clocks the bus masters directly .

Table continues on the next page...



Clock name	Description
Bus clock	System clock divided by OUTDIV4. Clocks the bus slaves and peripherals.
Flash clock	Flash memory clock On this device, it is the same as Bus clock.
MCGOUTCLK	MCG_Lite output of either IRC48M, IRC8M, MCG_Lite's external reference clock that sources the core, system, bus, and flash clock.
MCGIRCLK	IRC8M/2M internal reference clock divided by lirc_div2
MCGPCLK	MCG_Lite output of the fast (IRC48M) internal reference clock. This clock may clock some modules. In addition, this clock is used for LPUART0, LPUART1 and TPM modules
OSCCLK	System oscillator output of the internal oscillator or sourced directly from EXTAL. Used as MCG_Lite's external reference clock.
OSCERCLK	System oscillator output sourced from OSCCLK that may clock some on-chip modules
OSC32KCLK	System oscillator 32 kHz output
ERCLK32K	Clock source for some modules that is chosen as OSC32KCLK or RTC_CLKIN or LPO
LPO	PMC 1 kHz 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	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when...
MCGOUTCLK	Up to 48 MHz	Up to 8 MHz	MCG_Lite	In all stop modes except for partial stop modes.
MCGPCLK	Up to 48 MHz	Up to 48 MHz	MCG_Lite	MCG_Lite clock controls are not enabled and in all stop modes except for partial stop modes with MCG_MC[HIRCLPEN] cleared.
Core clock	Up to 48 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all wait and stop modes
Platform clock	Up to 48 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all stop modes
System clock	Up to 48 MHz	Up to 4 MHz	MCGOUTCLK clock divider	In all stop modes and Compute Operation

*Table continues on the next page...*

**Table 5-1. Clock summary (continued)**

Clock name	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when...
Bus clock	Up to 24 MHz	Up to 1 MHz <sup>2</sup>	MCGOUTCLK clock divider	In all stop modes except for partial STOP2 mode, and Compute Operation
SWD Clock	Up to 24 MHz	Up to 1 MHz	SWD_CLK pin	In all stop modes
Flash clock	Up to 24 MHz	Up to 1 MHz in EXT and LIRC	MCGOUTCLK clock divider	In all stop modes except for partial STOP2 mode
Internal reference (MCGIRCLK)	8/2MHz LIRC	8/2MHz LIRC	MCG_Lite	MCG_C1[IRCLKEN] cleared, Stop/VLPS mode and MCG_C1[IREFSTEN] cleared, or VLLS mode
External reference (OSCERCLK)	Up to 48 MHz (bypass), 30–40 kHz or 3-32Mhz(crystal)	Up to 16 MHz (bypass), 30–40 kHz (low-range crystal) or 3-16Mhz (high range crystal)	System OSC	System OSC's OSC_CR[ERCLKEN] cleared, or Stop mode and OSC_CR[EREFSTEN] cleared or VLLS0 and oscillator not in external clock mode.
External reference 32kHz (ERCLK32K)	30–40 kHz	30–40 kHz	System OSC or RTC_CLKIN or LPO	System OSC's OSC_CR[ERCLKEN] cleared and RTC's RTC_CR[OSCE] cleared or VLLS0 and oscillator not in external clock mode.
RTC_CLKOUT	RTC 1Hz, OSCERCLK	RTC 1Hz, OSCERCLK	RTC 1Hz, OSCERCLK	Clock is disabled in VLLSx modes
CLKOUT32K	32K	32K	ERCLK32K	SIM_SOPT1[OSC32KOUT] not configured to drive ERCLK32K out.
LPO	1 kHz	1 kHz	PMC	in VLLS0
TPM clock	Up to 48 MHz	Up to 48 MHz <sup>1</sup>	MCGIRCLK, MCGPCLK, OSCERCLK	SIM_SOPT2[TPMSRC]=00 selected clock source disabled
LPUART0 clock	Up to 48 MHz	Up to 48 MHz <sup>1</sup>	MCGIRCLK, MCGPCLK, OSCERCLK	SIM_SOPT2[LPUART0SRC]=00 selected clock source disabled

Table continues on the next page...

**Table 5-1. Clock summary (continued)**

Clock name	Run mode clock frequency	VLPR mode clock frequency	Clock source	Clock is disabled when...
LPUART1 clock	Up to 48 MHz	Up to 48 MHz <sup>1</sup>	MCGIRCLK, MCGPCLK, OSCERCLK	SIM_SOPT2[LPUART1 SRC]=00 selected clock source disabled
FlexIO clock	Up to 48 MHz	Up to 48 MHz <sup>1</sup>	MCGIRCLK, MCGPCLK, OSCERCLK	SIM_SOPT2[FLEXIOS RC]=00 selected clock source disabled

1. Before entering the VLPR mode, besides the HIRCEN and HIRCLPEN bits, the BGEN and bit 6 VLPO should be set.
2. If in LIRC mode, where clocking is derived from the fast internal reference clock, the Bus clock and flash clock frequency needs to be limited to 1Mhz if executing from flash.

## 5.5 Internal clocking requirements

The clock dividers are programmed via the SIM\_CLKDIV1 register. The following requirements must be met when configuring the clocks for this device:

- The core, platform, and system clock are programmable from a divide-by-1 through divide-by-16 setting. The core, platform, and system clock frequencies must be 48 MHz or slower.
- The frequency of bus clock and flash clock is divided by the system clock and is programmable from a divide-by-1 through divide-by-8 setting. The bus clock and flash clock must be programmed to 24 MHz or slower.
- MCGPCLK is used for peripheral which is fixed to 48 MHz.
- MCGIRCLK is also one of peripheral clock sources which is from IRC8M and can be divided down by a divider.

The following is a common clock configuration for this device:

Clock	Max. Frequency
Core clock	48 MHz
Platform clock	48 MHz
System clock	48 MHz
Bus clock	24 MHz
Flash clock	24 MHz
MCGIRCLK	8 MHz
MCGPCLK	48 MHz

### 5.5.1 Clock divider values after reset

Each clock divider is programmed via the CLKDIV1 registers of the SIM module. Two bits in the flash memory's FTFA\_FOPT register control the reset value of the core clock, system clock, bus clock, and flash clock dividers as shown in the table given below:

FTFA_FOPT [4,0]	Core/system clock	Bus/Flash clock	Execution Mode
00	0x7 (divide by 8)	0x1 (divide by 2)	VLPR
01	0x3 (divide by 4)	0x1 (divide by 2)	VLPR
10	0x1 (divide by 2)	0x1 (divide by 2)	RUN
11	0x0 (divide by 1)	0x1 (divide by 2)	RUN

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

The flash erased state defaults to fast clocking mode, since these bits reside in flash, which is logic 1 in the flash erased state. To enable a lower power boot option, program the appropriate bits in FTFA\_FOPT. During the reset sequence, if either of the control bits is cleared, the system is in a slower clock configuration. Upon any system reset, the clock dividers return to this configurable reset state.

The default reset clock for core/system clock is 8 MHz from IRC8M.

### 5.5.2 VLPR mode clocking

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

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

## 5.6 Clock gating

The clock to each module can be individually gated on and off using bits of the SCGCx registers of the SIM module. These bits are cleared after any reset, which disables the clock to the corresponding module to conserve power. Prior to initializing a module, set the corresponding bit in the 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-M0+ core	Platform clock	Core clock	—
NVIC	Platform clock	—	—
DAP	Platform clock	—	SWD_CLK
<b>System modules</b>			
DMA	System clock	—	—
DMA Mux	Bus clock	—	—
Port control	Bus clock	—	—
Crossbar Switch	Platform clock	—	—
Peripheral bridges	System clock	Bus clock	—
LLWU, PMC, SIM, RCM	Bus clock	LPO	—
Mode controller	Bus clock	—	—
MCM	Platform clock	—	—
COP watchdog	Bus clock	LPO, Bus Clock, MCGIRCLK, OSCERCLK	—
CRC	Bus clock	—	—
<b>Clocks</b>			
MCG_Lite	Bus clock	MCGOUTCLK, MCGPCLK, MCGIRCLK, OSCERCLK, ERCLK32K	—
OSC	Bus clock	OSCERCLK	—
<b>Memory and memory interfaces</b>			
Flash Controller	Platform clock	Flash clock	—
Flash memory	Flash clock	—	—
<b>Analog</b>			
ADC	Bus clock	OSCERCLK	—
CMP	Bus clock	—	—
DAC	Bus clock	—	—
Internal Voltage Reference (VREF)	Bus clock	—	—
<b>Timers</b>			
TPM	Bus clock	TPM clock	TPM_CLKIN0, TPM_CLKIN1
PIT	Bus clock	—	—
LPTMR	Bus clock	LPO, OSCERCLK, MCGPCLK, ERCLK32K	—

Table continues on the next page...

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

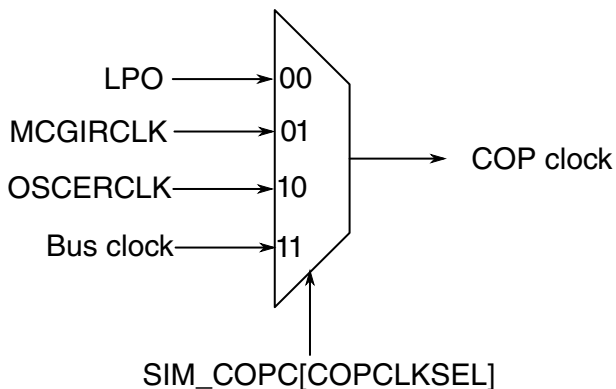
Module	Bus interface clock	Internal clocks	I/O interface clocks
RTC	Bus clock	ERCLK32K	RTC_CLKOUT, RTC_CLKIN
<b>Communication interfaces</b>			
SPI0	Bus clock	—	SPI0_SCK
SPI1	System clock	—	SPI1_SCK
I <sup>2</sup> C0	System Clock	—	I2C0_SCL
I <sup>2</sup> C1	System Clock	—	I2C1_SCL
LPUART0, LPUART1	Bus clock	LPUART0 clock LPUART1 clock	—
UART2	Bus clock	—	—
FlexIO	Bus clock	FlexIO 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 COP clocking

The COP may be clocked from four clock sources as shown in the following figure.



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

### 5.7.3 RTC clocking

The RTC module can be clocked as shown in the following figure.

#### NOTE

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

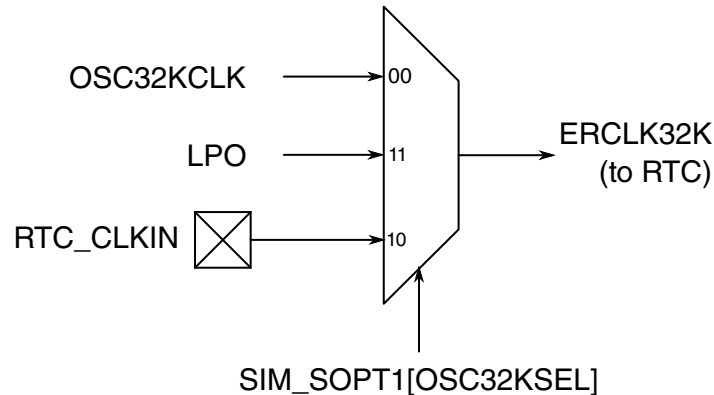


Figure 5-3. RTC clock generation

### 5.7.4 RTC\_CLKOUT and CLKOUT32K clocking

When the RTC is enabled and the port control module selects the RTC\_CLKOUT function, the RTC\_CLKOUT signal, controlled from SIM\_SOPT2[RTCCLKOUTSEL], outputs a 1 Hz or 32 kHz output derived from RTC oscillator as shown below and can be configured to drive to external pins via pin control configuration for the associated pin. It is also possible to drive CLKOUT32K on the same pins as controlled by SIM\_SOPT1[OSC32KOUT] on the selected RTC\_CLKOUT pins in all modes of operation (including LLS/VLLS and System Reset), overriding the existing pin mux configuration for that pin.

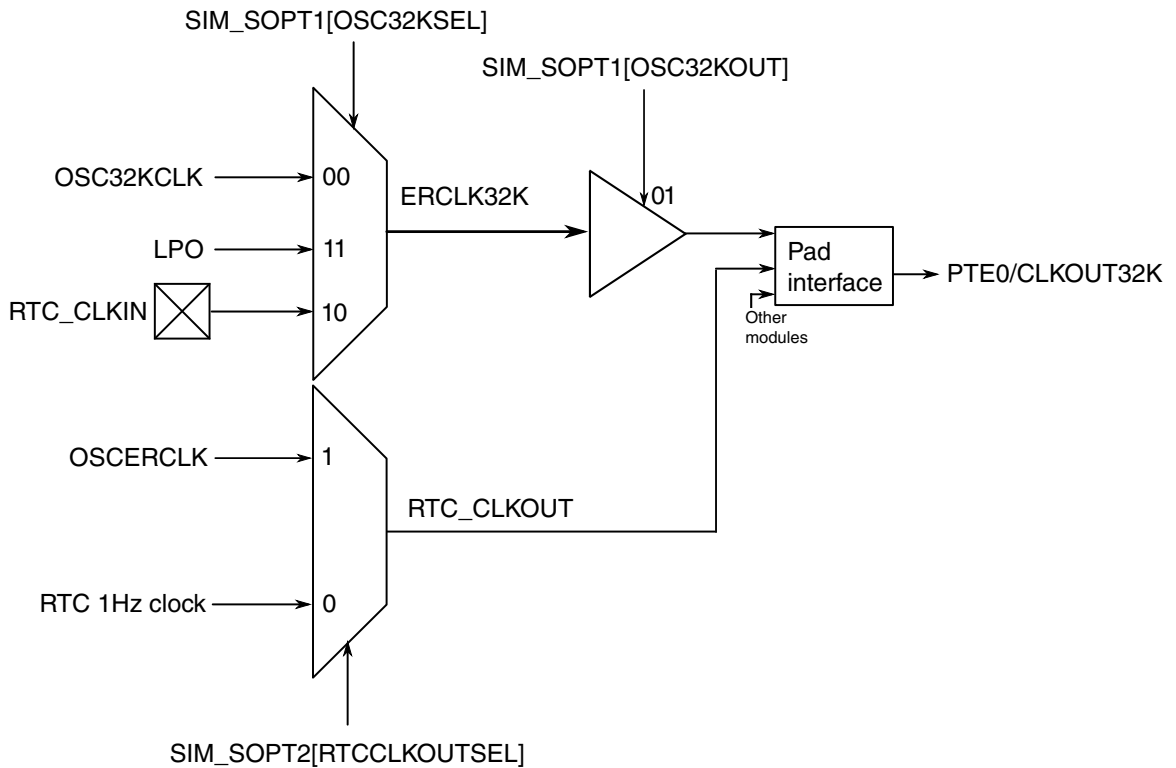


Figure 5-4. RTC\_CLKOUT and CLKOUT32K generation

### 5.7.5 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.

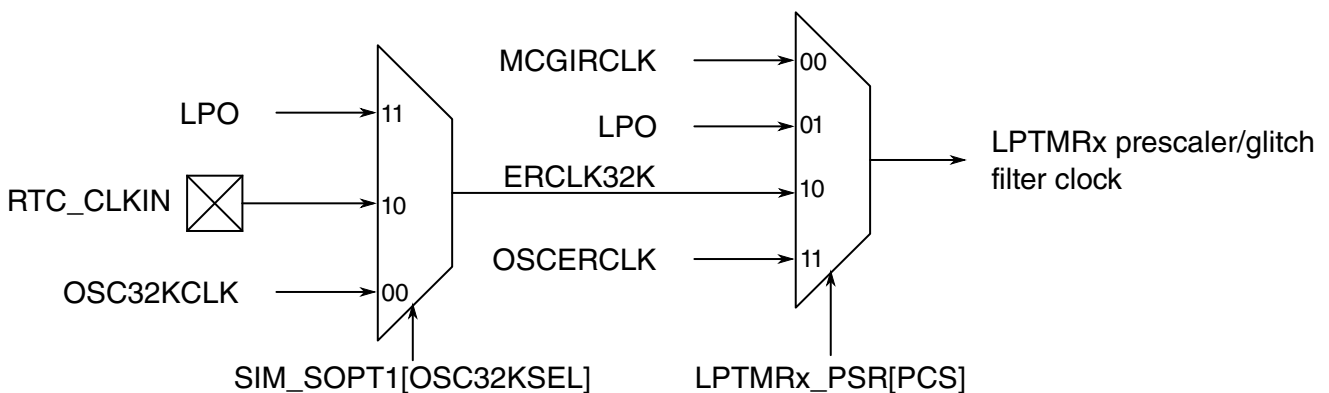


Figure 5-5. LPTMR<sub>x</sub> prescaler/glitch filter clock generation



## 5.7.6 TPM clocking

The counter for the TPM modules has a selectable clock as shown in the following figure.

### NOTE

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

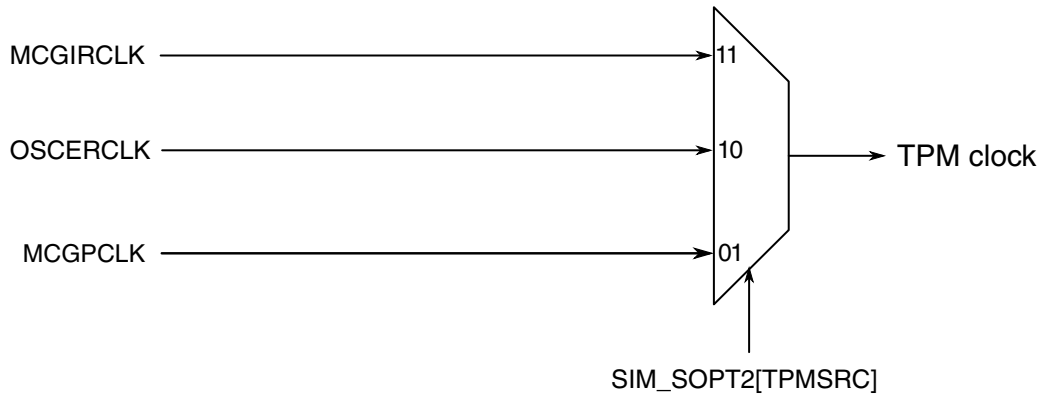


Figure 5-6. TPM clock generation

## 5.7.7 LPUART clocking

The LPUART0 and LPUART1 have a selectable clock as shown in the following figure. UART2 module operates from the bus clock.

### NOTE

The chosen clock must remain enabled if the LPUART0 and LPUART1 is to continue operating in all required low-power modes.

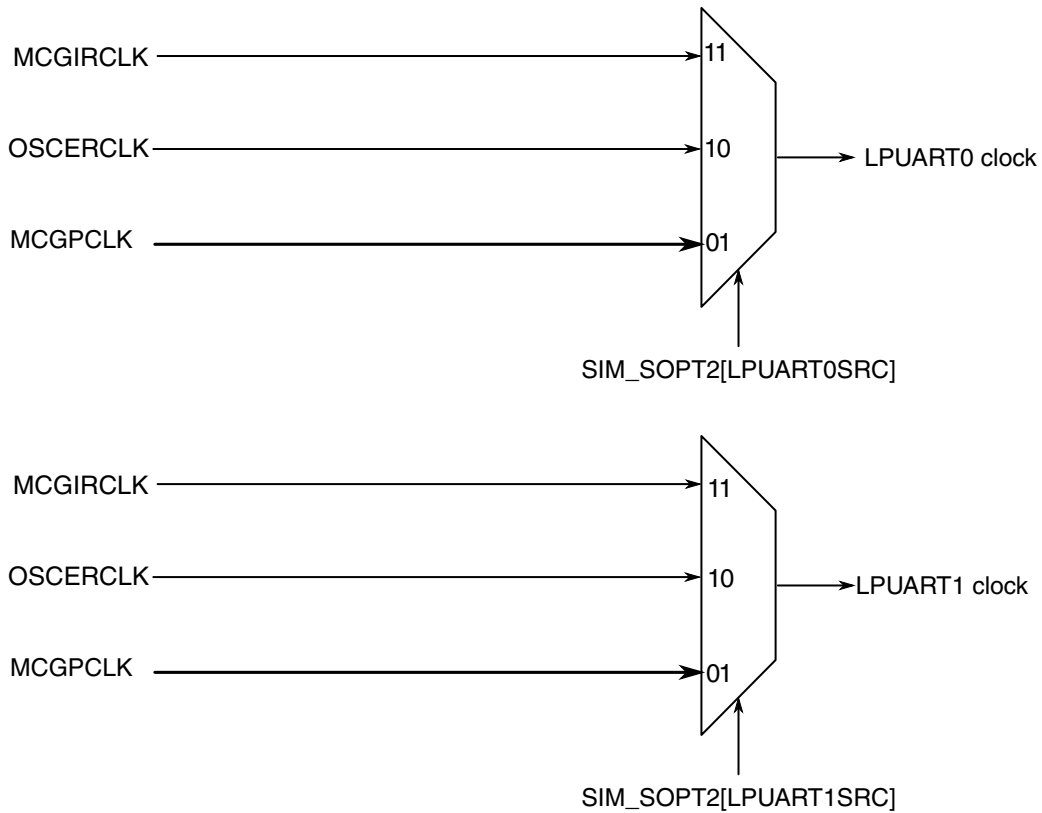


Figure 5-7. LPUART0 and LPUART1 clock generation

### 5.7.8 FlexIO clocking

The FlexIO module has a selectable clock as shown in the following figure.

**NOTE**

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

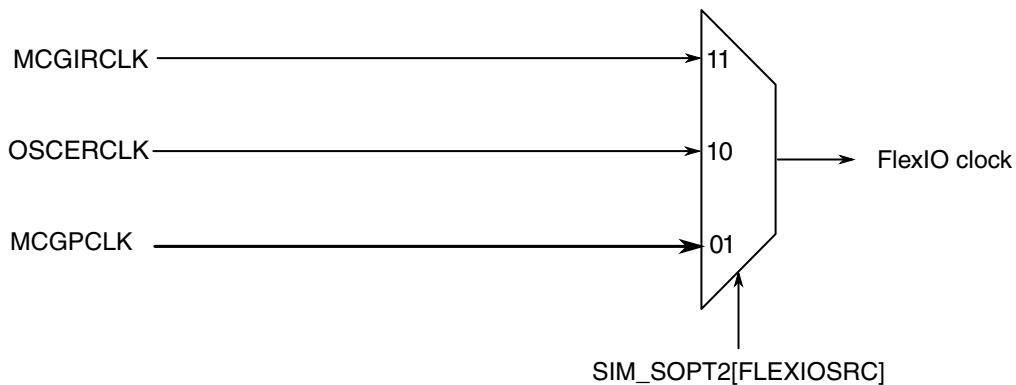


Figure 5-8. FlexIO clock generation

# Chapter 6

## Reset and Boot

### 6.1 Introduction

The reset sources supported in this MCU are listed in the table found here.

**Table 6-1. Reset sources**

Reset sources	Description
POR reset	<ul style="list-style-type: none"><li>• <a href="#">Power-on reset (POR)</a></li></ul>
System resets	<ul style="list-style-type: none"><li>• <a href="#">External pin reset (PIN)</a></li><li>• <a href="#">Low-voltage detect (LVD)</a></li><li>• <a href="#">Computer operating properly (COP) watchdog reset</a></li><li>• <a href="#">Low leakage wakeup (LLWU) reset</a></li><li>• <a href="#">Stop mode acknowledge error (SACKERR)</a></li><li>• <a href="#">Software reset (SW)</a></li><li>• <a href="#">Lockup reset (LOCKUP)</a></li><li>• <a href="#">MDM DAP system reset</a></li></ul>
Debug reset	<ul style="list-style-type: none"><li>• <a href="#">Debug reset</a></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 can exit and reset in functional mode where the CPU is executing code (default) or the CPU is in a debug halted state. There are several boot options that can be configured. See [Boot information](#) for more details.

### 6.2 Reset

The information found here 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 fields in the Reset Status Register are set following a POR.

## 6.2.2 System reset sources

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

- Reads the start SP (SP\_main) from vector-table offset 0
- Reads the start program counter (PC) from vector-table offset 4
- Link register (LR) is set to 0xFFFF\_FFFF.

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

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

- SWD\_CLK in pulldown (PD)
- SWD\_DIO in pullup (PU)

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

This pin is pseudo open drain to be compatible with other L family and has an internal pullup device. Asserting  $\overline{RESET}$  wakes the device from any mode.

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

### 6.2.2.1.1 $\overline{\text{RESET}}$ 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, and VLLS1 with SMC\_STOPCTRL[LPOPO]=0), 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 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 PMC\_LVDSC1[LVDRE] to 1. The low-voltage detection threshold is determined by PMC\_LVDSC1[LVDV]. 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. RCM\_SRS0[LVD] 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 RCM\_SRS0[WDOG] to set.

### 6.2.2.4 Low leakage wakeup (LLWU)

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

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

#### NOTE

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

### 6.2.2.5 Stop mode acknowledge error (SACKERR)

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

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

### 6.2.2.6 Software reset (SW)

The SYSRESETREQ field 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 RCM\_SRS1[SW] to set.

### 6.2.2.7 Lockup reset (LOCKUP)

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

The LOCKUP condition causes a system reset and also causes RCM\_SRS1[LOCKUP] to set.

### 6.2.2.8 MDM-AP system reset request

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

Set the Core Hold Reset field 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-Lite.

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

### 6.2.3.4 Chip Reset not VLLS

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

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

### 6.2.3.5 Early Chip Reset

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

### 6.2.3.6 Chip Reset

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

## 6.2.4 $\overline{\text{RESET}}$ pin

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

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

The  $\overline{\text{RESET}}$  pin can be disabled by programming FTFA\_FOPT[RESET\_PIN\_CFG] option bit to 0 (See [Table 6-2](#)). When this option is selected, there could be a short period of contention during a POR ramp where the device drives the pinout low prior to establishing the setting of this option and releasing the reset function on the pin.

## 6.3 Boot

The information found here describes the boot sequence, including sources and options.



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

### 6.3.1 Boot sources

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

- internal flash
- ROM

This device supports booting from internal flash with the reset vectors located at addresses 0x0 (initial SP\_main), 0x4 (initial PC).

The device is also able to boot from ROM. The ROM start address is from 0x1C00\_0000. When boot from ROM, it remaps all vector fetch to ROM base address. ROM code start pointer locates in ROM vector table which address is 0x1C00\_0000 where stack pointer is offset 0x0 and reset vector is offset 0x4. Vector table and stack pointer are valid out of reset. RCM mode register is cleared by software when Boot ROM completes, this disables remapping of vector fetches. Boot source can change between reset, but is always known before core reset negation.  $\overline{\text{NMI}}$  input is disabled to platform when booting from ROM. See FOPT section and [Reset Control Module](#) for more detail options.

The boot options can be overridden by using RCM\_FM[2:1] and RCM\_MR[2:1] which can be written by software. The boot source remains set until the next System Reset or software can write logic one to clear one or both of the mode bits.

### 6.3.2 FOPT boot options

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

The MCU uses FTFA\_FOPT to configure the device at reset as shown in the following table. An FTFA\_FOPT value of 0x00 is invalid and will be ignored. The FOPT register is written to 0xFF if the contents of the Flash nonvolatile option are 0x00.

**Table 6-2. Flash Option Register (FTFA\_FOPT) definition**

Bit Num	Field	Value	Definition
7-6	BOOTSRC_SEL		Boot Source Selection: these bits select the boot sources if boot pin option bit BOOTPIN_OPT = 1
		00	Boot from Flash
		01	Reserved
		10	Boot from ROM
		11	Boot from ROM
5	FAST_INIT		Selects 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.
3	RESET_PIN_CFG		Enables/disables control for the $\overline{\text{RESET}}$ pin.
		0	$\overline{\text{RESET}}$ pin is disabled following a POR and cannot be enabled as reset function. When this option is selected, there could be a short period of contention during a POR ramp where the device drives the pinout low prior to establishing the setting of this option and releasing the reset function on the pin.  This bit is preserved through system resets and low-power modes. When $\overline{\text{RESET}}$ pin function is disabled, it cannot be used as a source for low-power mode wake-up.  <b>NOTE:</b> When the reset pin has been disabled and security has been enabled by means of the FSEC register, a mass erase can be performed only by setting both the Mass Erase and System Reset Request fields in the MDM-AP register.
		1	RESET_b pin is dedicated. The port is configured as open drain and pullup enabled.
2	NMI_DIS		Enables/disables control for the NMI function.
		0	NMI interrupts are always blocked. The associated pin continues to default to $\overline{\text{NMI}}$ pin controls with internal pullup enabled. When $\overline{\text{NMI}}$ pin function is disabled, it cannot be used as a source for low-power mode wake-up.
		1	NMI_b pin/interrupts reset default to enabled.
1	BOOTPIN_OPT		External pin selects boot options
		0	Force Boot from ROM if BOOTCFG0 asserted, where BOOTCFG0 is the boot config function which is muxed with $\overline{\text{NMI}}$ pin. $\overline{\text{RESET}}$ pin must be enabled when this option is selected.
		1	Boot source configured by FOPT[7:6] ( BOOTSRC_SEL) bits
4,0	LPBOOT		Controls the reset value of OUTDIV1 value in SIM_CLKDIV1 register, and the state of the RUNM register in SMC_PMCTRL. 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. <sup>1</sup>

Table continues on the next page...

**Table 6-2. Flash Option Register (FTFA\_FOPT) definition  
(continued)**

Bit Num	Field	Value	Definition
		00	Core and system clock divider (OUTDIV1) is 0x7 (divide by 8). Device is configured for VLPR mode on exit from reset.
		01	Core and system clock divider (OUTDIV1) is 0x3 (divide by 4). Device is configured for VLPR mode on exit from reset.
		10	Core and system clock divider (OUTDIV1) is 0x1 (divide by 2). Device is configured for RUN mode on exit from reset.
		11	Core and system clock divider (OUTDIV1) is 0x0 (divide by 1). Device is configured for RUN mode on exit from reset.

1. Refer to [Clock divider values after reset](#) and RCM\_FM, RCM\_MR in the [Reset Control Module \(RCM\)](#) for details.

### 6.3.3 Boot sequence

At power up, the on-chip regulator holds the system in a POR state until the input supply exceeds the POR threshold. The system continues to be held in this static state until the internally regulated supplies have reached a safe operating voltage as determined by the LVD. The Reset Controller logic then controls a sequence to exit reset.

1. A system reset is held on internal logic, the  $\overline{\text{RESET}}$  pin is driven out low, and the MCG-Lite is enabled in its default clocking mode.
2. Required clocks are enabled (system clock, flash clock, and any bus clocks that do not have clock gate control reset to disabled).
3. The system reset on internal logic continues to be held, but the Flash Controller is released from reset and begins initialization operation while the Reset Control logic continues to drive the  $\overline{\text{RESET}}$  pin out low.
4. Early in reset sequencing, the NVM option byte is read and stored to the FOPT register of the Flash Memory module (FTFA\_FOPT). If the bits associated with FTFA\_FOPT[LPBOOT] are programmed for an alternate clock divider reset value, the system/core clock is switched to a slower clock speed. If FTFA\_FOPT[FAST\_INIT] 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. The next sequence of events depends on the  $\overline{\text{NMI}}/\text{BOOTCFG0}$  input

and FTFA\_FOPT[NMI\_DIS] and FTFA\_FOPT[BOOTSRC\_SEL] and FTFA\_FOPT[BOOTPIN\_OPT] as well as RCM\_FM[FORCEROM] and RCM\_MR[BOOTROM](See [Table 6-2](#) and RCM block guide) :

- If the  $\overline{\text{NMI}}/\text{BOOTCFG0}$  input is high or the NMI function is disabled in FTFA\_FOPT, the CPU begins execution at the PC location.
- If the  $\overline{\text{NMI}}/\text{BOOTCFG0}$  input is low, the NMI function is enabled in FTFA\_FOPT, and FTFA\_FOPT[BOOTPIN\_OPT] = 1, 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.
- When FTFA\_FOPT[BOOTPIN\_OPT] = 0, it forces boot from ROM if  $\overline{\text{NMI}}/\text{BOOTCFG0}$  pin set to 0.

### NOTE

If the NMI function is not required, either for an interrupt or wake up source, it is recommended that the NMI function be disabled by clearing NMI\_DIS in the FOPT register.

Subsequent system resets follow this same reset flow.

# Chapter 7

## Power Management

### 7.1 Introduction

Information about the various chip power modes and functionality of the individual modules in these modes can be found here.

See [AN4503: Power Management for Kinetis and ColdFire+ MCUs](#) for further details on power management techniques.

### 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 either Run mode or VLP Run mode.

**NOTE**

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

Because Compute Operation reuses the Stop mode logic (including the staged entry with bus masters disabled before bus slaves), any bus master or bus slave that can remain functional in Stop mode also remains functional in Compute Operation, including generation of asynchronous interrupts and DMA requests. When enabling Compute Operation in Run mode, module functionality for bus masters and slaves is the equivalent of STOP mode. When enabling Compute Operation in VLP Run mode, module functionality for bus masters and slaves is the equivalent of VLPS mode. The MCG, PMC, SRAM, and Flash read port are not affected by Compute Operation, although the Flash register interface is disabled.

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

Compute Operation is controlled by the CPO register in the MCM (MCM\_CPO), which is only accessible to the CPU. Setting or clearing MCM\_CPO[CPOREQ] 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 field in the CPO register of MCM module (MCM\_CPO[CPOACK]) indicates when entry has completed.
- When exiting Compute Operation in Run mode, MCM\_CPO[CPOACK] 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 that MCM\_CPO[CPOACK] is polled to determine when the AIPS peripheral space can be accessed without generating a bus error.

The DMA wake-up is also supported during Compute Operation and causes MCM\_CPO[CPOACK] to clear and the AIPS peripheral space to be accessible for the duration of the DMA wakeup. At the completion of the DMA wake-up, 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. The bits of these registers 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, see the [Clock Distribution](#) and [SIM](#) chapters.

## 7.3 Power modes

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

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

For each run mode, there is a corresponding Wait and Stop mode. Wait modes are similar to ARM Sleep modes. Stop modes (VLPS, STOP) are similar to ARM Sleep Deep mode. The Very Low Power Run (VLPR) operating mode can drastically reduce runtime power when the maximum bus frequency is not required to handle the application needs.

## Power modes

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

**Table 7-1. Chip power modes**

Chip mode	Description	Core mode	Normal recovery method
Normal Run	Allows maximum performance of chip. <ul style="list-style-type: none"> <li>• Default mode out of reset</li> <li>• On-chip voltage regulator is on.</li> </ul>	Run	—
Normal Wait - via WFI	Allows peripherals to function while the core is in Sleep mode, reducing power. <ul style="list-style-type: none"> <li>• NVIC remains sensitive to interrupts</li> <li>• Peripherals continue to be clocked.</li> </ul>	Sleep	Interrupt
Normal Stop - via WFI	Places chip in static state. Lowest power mode that retains all registers while maintaining LVD protection. <ul style="list-style-type: none"> <li>• NVIC is disabled.</li> <li>• AWIC is used to wake up from interrupt.</li> <li>• Peripheral clocks are stopped.</li> </ul>	Sleep Deep	Interrupt
VLPR (Very Low-Power Run)	On-chip voltage regulator is in a low-power mode that supplies only enough power to run the chip at a reduced frequency. Only MCG-Lite modes LIRC and EXT can be used in VLPR. <ul style="list-style-type: none"> <li>• Reduced frequency Flash access mode (1 MHz)</li> <li>• LVD off</li> <li>• In LIRC clock mode, only the internal reference oscillator (LIRC8M) is available to provide a low power nominal 4 MHz source for the core with the nominal bus and flash clock required to be &lt;1 MHz</li> <li>• Alternatively, EXT clock mode can be used with an external clock or the crystal oscillator providing the clock source.</li> </ul>	Run	—
VLPW (Very Low-Power Wait) -via WFI	Same as VLPR but with the core in Sleep mode to further reduce power. <ul style="list-style-type: none"> <li>• NVIC remains sensitive to interrupts (CPU clk = ON).</li> <li>• On-chip voltage regulator is in a low-power mode that supplies only enough power to run the chip at a reduced frequency.</li> </ul>	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. <ul style="list-style-type: none"> <li>• Peripheral clocks are stopped, but OSC, LPTMR, RTC, CMP can be used.</li> <li>• UART, LPUART and TPM can optionally be enabled if their clock source is enabled.</li> <li>• NVIC is disabled (CPU clk = OFF); AWIC is used to wake up from interrupt.</li> <li>• On-chip voltage regulator is in a low-power mode that supplies only enough power to run the chip at a reduced frequency.</li> <li>• All SRAM is operating (content retained and I/O states held).</li> </ul>	Sleep Deep	Interrupt
LLS (Low-Leakage Stop)	State retention power mode <ul style="list-style-type: none"> <li>• Most peripherals are in state retention mode (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP can be used.</li> <li>• NVIC is disabled; LLWU is used to wake up.</li> </ul>	Sleep Deep	Wake-up Interrupt <sup>1</sup>

*Table continues on the next page...*

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

Chip mode	Description	Core mode	Normal recovery method
	<p><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</p> <ul style="list-style-type: none"> <li>All SRAM is operating (content retained and I/O states held).</li> </ul>		
VLLS3 (Very Low-Leakage Stop3)	<ul style="list-style-type: none"> <li>Most peripherals are disabled (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP can be used.</li> <li>NVIC is disabled; LLWU is used to wake up.</li> <li>SRAM_U and SRAM_L remain powered on (content retained and I/O states held).</li> </ul>	Sleep Deep	Wake-up Reset <sup>2</sup>
VLLS1 (Very Low-Leakage Stop1)	<ul style="list-style-type: none"> <li>Most peripherals are disabled (with clocks stopped), but OSC, LLWU, LPTMR, RTC, CMP can be used.</li> <li>NVIC is disabled; LLWU is used to wake up.</li> <li>All of SRAM_U and SRAM_L are powered off.</li> <li>The 32-byte system register file remains powered for customer-critical data</li> </ul>	Sleep Deep	Wake-up Reset <sup>2</sup>
VLLS0 (Very Low-Leakage Stop 0)	<ul style="list-style-type: none"> <li>Most peripherals are disabled (with clocks stopped), but LLWU, LPTMR, RTC can be used.</li> <li>NVIC is disabled; LLWU is used to wake up.</li> <li>All of SRAM_U and SRAM_L are powered off.</li> <li>The 32-byte system register file remains powered for customer-critical data</li> <li>LPO disabled, optional POR brown-out detection</li> </ul>	Sleep Deep	Wake-up Reset <sup>2</sup>

- Resumes Normal Run mode operation by executing the LLWU interrupt service routine.
- Follows the reset flow with the LLWU interrupt flag set for the NVIC.

## 7.4 Entering and exiting power modes

The WFI instruction invokes wait and stop modes for the chip. The processor exits the low-power mode via an interrupt.

For LLS and VLLS modes, the wake-up sources are limited to LLWU generated wake-ups,  $\overline{\text{NMI}}_b$  pin, or  $\overline{\text{RESET}}_b$  pin assertions. When the  $\overline{\text{NMI}}_b$  pin or  $\overline{\text{RESET}}_b$  pin have been disabled through associated FTFA\_FOPT settings, then these pins are ignored as wakeup sources. The wake-up flow from VLLSx is always through reset.

### NOTE

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

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

## 7.5 Module operation in low-power modes

The table found here illustrates the functionality of each module while the chip is in each of the low power modes.

The standard behavior is shown with some exceptions for Compute Operation (CPO) and Partial Stop2 (PSTOP2).

Debug modules are discussed separately; see [Debug in low-power modes](#). Number ratings (such as 4 MHz and 1 Mbit/s) represent the maximum frequencies or maximum data rates per mode. Following is list of terms also used in the table.

- 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 wake-up. For clocks, OFF means disabled.
- wakeup = Modules can serve as a wake-up source for the chip.

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

Modules	VLPR	VLPW	Stop	VLPS	LLS	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 VLLS3, OFF in VLLS0/1
LVD	disabled	disabled	ON	disabled	disabled	disabled
Brown-out Detection	ON	ON	ON	ON	ON	ON in VLLS1/3, optionally disabled in VLLS0 <sup>3</sup>
DMA	FF Async operation in CPO	FF	Async operation	Async operation	static	OFF

*Table continues on the next page...*

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

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
COP watchdog	FF static in CPO	FF	Optional work with clock source enabled in stop mode FF in PSTOP2	Optional work with clock source enabled in stop mode	static	OFF
CRC	FF	FF	static	static	Static	OFF
Clocks						
1kHz LPO	ON	ON	ON	ON	ON	ON in VLLS1/3, optional be disabled by SMC_STOPCTL [LPOPO] , OFF in VLLS0
System oscillator (OSC)	OSCERCLK max of 16MHz crystal	OSCERCLK max of 16MHz crystal	OSCERCLK optional	OSCERCLK max of 16MHz crystal	OSCERCLK max of 16MHz crystal	OSCERCLK max of 16MHz crystal in VLLS1/3, OFF in VLLS0
MCG_Lite	4 MHz	4 MHz	static - MCGIRCLK optional, MCGP CLK optional	static - MCGIRCLK optional	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
Bus clock	1 MHz max OFF in CPO	1 MHz max	OFF  24 MHz max in PSTOP2 from RUN  1 MHz max in PSTOP2 from VLPR	OFF	OFF	OFF
Memory and memory interfaces						
Flash	1 MHz max access - no program  No register access in CPO	low power	low power	low power	OFF	OFF
SRAM_U and SRAM_L	low power	low power	low power	low power	low power	low power in VLLS3, OFF in VLLS0/1
System Register File	powered	powered	powered	powered	powered	powered
Communication interfaces						
LPUART0, LPUART1	1 Mbit/s	1 Mbit/s	Async operation	Async operation	static	OFF

Table continues on the next page...

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

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
	Async operation in CPO		FF in PSTOP2			
UART2	62.5 kbit/s static, wakeup on edge in CPO	62.5 kbit/s	static, wakeup on edge FF in PSTOP2	static, wakeup on edge	static	OFF
SPI0 (without FIFO)	master mode 500 kbit/s, slave mode 250 kbit/s static, slave mode receive in CPO	master mode 500 kbit/s, slave mode 250 kbit/s	static, slave mode receive FF in PSTOP2	static, slave mode receive	static	OFF
SPI1 (with FIFO)	master mode 2 Mbit/s, slave mode 1 Mbit/s static, slave mode receive in CPO	master mode 2 Mbit/s, slave mode 1 Mbit/s	static, slave mode receive	static, slave mode receive	static	OFF
I <sup>2</sup> C0	100 kbit/s static, address match wakeup in CPO	100 kbit/s	static, address match wakeup	static, address match wakeup	static	OFF
I <sup>2</sup> C1	100 kbit/s static, address match wakeup in CPO	100 kbit/s	static, address match wakeup	static, address match wakeup	static	OFF
FlexIO	FF	FF	FF	FF	static	OFF
<b>Timers</b>						
TPM	FF Async operation in CPO	FF	Async operation FF in PSTOP2	Async operation	static	OFF
PIT	FF static in CPO	FF	static	static	static	OFF
LPTMR	FF	FF	Async operation FF in PSTOP2	Async operation	Async operation	Async operation <sup>4</sup>
RTC	FF Async operation in CPO	FF	Async operation FF in PSTOP2	Async operation	Async operation	Async operation <sup>5</sup>
<b>Analog</b>						
16-bit ADC	FF ADC internal clock only in CPO	FF	ADC internal clock only FF in PSTOP2	ADC internal clock only	static	OFF

Table continues on the next page...

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

Modules	VLPR	VLPW	Stop	VLPS	LLS	VLLSx
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 VLLS1/3, OFF in VLLS0
6-bit DAC	FF static in CPO	FF	static FF in PSTOP2	static	static	static, OFF in VLLS0
12-bit DAC	FF static in CPO	FF	static FF in PSTOP2	static	static	static, OFF in VLLS0
Internal Voltage Reference (VREF)	FF static in CPO	FF	static FF in PSTOP2	static	static	static, OFF in VLLS0
<b>Human-machine interfaces</b>						
GPIO	FF IOPORT 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.
- STOPCTRL[PORPO] in the SMC module controls this option.
- LPO clock source is not available in VLLS0. Also, to use system OSC in VLLS0 it must be configured for bypass (external clock) operation. Pulse counting is available in all modes.
- In VLLS0 the only clocking option is from RTC\_CLKIN.
- CMP in stop or VLPS supports high speed or low speed external pin to pin or external pin to DAC compares. CMP in LLS or VLLSx only supports low speed external pin to pin or external pin to DAC compares.





# 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.

#### 8.1.1 Flash security

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

#### NOTE

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

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

#### 8.1.2 Security interactions with other modules

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

### **8.1.2.1 Security interactions with Debug**

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

Although most debug functions are disabled, the debugger can write to the Flash Mass Erase in Progress field of 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 debug of this device is based on the ARM CoreSight™ architecture and is configured to provide the maximum flexibility as allowed by the restrictions of the pinout and other available resources.

It provides register and memory accessibility from the external debugger interface, basic run/halt control plus 2 breakpoints and 2 watchpoints.

Only one debug interface is supported:

- Serial Wire Debug (SWD)

### 9.2 Debug port pin descriptions

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

**Table 9-1. Serial wire debug pin description**

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

### 9.3 SWD status and control registers

Through the ARM Debug Access Port (DAP), the debugger has access to the status and control elements, implemented as registers on the DAP bus as shown in the figure found here.

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 using SWD. The MDM-AP is accessible as Debug Access Port 1 with the available registers shown in this table.

**Table 9-2. MDM-AP register summary**

Address	Register	Description
0x0100_0000	Status	See <a href="#">MDM-AP Status Register</a>
0x0100_0004	Control	See <a href="#">MDM-AP Control Register</a>
0x0100_00FC	IDR	Read-only identification register that always reads as 0x001C_0020

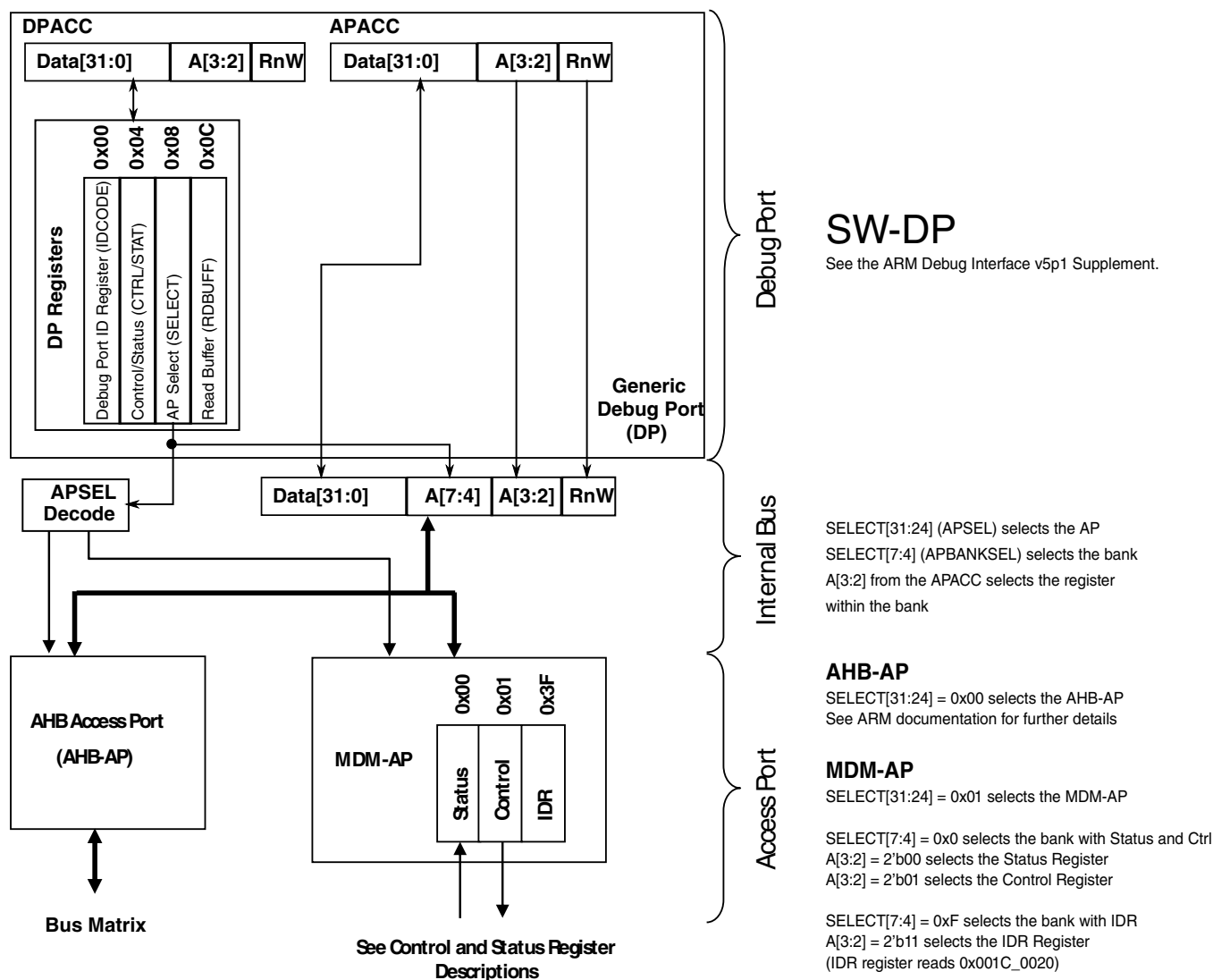


Figure 9-1. MDM AP addressing

### 9.3.1 MDM-AP Control Register

Table 9-3. MDM-AP Control register assignments

Bit	Name	Secure <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.

Table continues on the next page...

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

Bit	Name	Secure <sup>1</sup>	Description
			If the core is in a Stop or Wait mode, this bit can be used to wake the core and transition to a halted state.
3	System Reset Request	Y	Set to force a system reset. The system remains held in reset until this bit is cleared.
4	Core Hold Reset	N	Configuration bit to control core operation at the end of system reset sequencing.  0 Normal operation: Release the core from reset along with the rest of the system at the end of system reset sequencing.  1 Suspend operation: Hold the core in reset at the end of reset sequencing. 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 is ignored on a VLLS wakeup via the Reset pin. During a VLLS wakeup via the Reset pin, the system can be held in reset by holding the reset pin asserted allowing the debugger to reinitialize the debug modules.  This bit holds the system in reset when VLLSx modes are exited to allow the debugger time to 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 system in reset until VLLDBGACK is asserted.  VLLDBGREQ clears automatically due to the POR reset generated as part of the VLLSx recovery.
6	VLLSx Debug Acknowledge (VLLDBGACK)	N	Set to release a system being held in reset following a VLLSx recovery  This bit is used by the debugger to release the system reset when it is being held on VLLSx mode exit. The debugger re-initializes all debug IP and then assert this control bit to allow the Mode Controller to release the system from reset and allow CPU operation to begin.  VLLDBGACK 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.
8 – 31	Reserved for future use	N	

1. Command available in secure mode

## 9.3.2 MDM-AP Status Register

**Table 9-4. MDM-AP Status register assignments**

Bit	Name	Description
0	Flash Mass Erase Acknowledge	<p>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.</p> <p>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.</p>
1	Flash Ready	Indicates 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	<p>Indicates the system reset state.</p> <p>0 System is in reset.</p> <p>1 System is not in reset.</p>
4	Reserved	
5	Mass Erase Enable	<p>Indicates if the MCU can be mass erased or not</p> <p>0 Mass erase is disabled.</p> <p>1 Mass erase is enabled .</p>
6	Backdoor Access Key Enable	<p>Indicates if the MCU has the backdoor access key enabled.</p> <p>0 Disabled</p> <p>1 Enabled</p>
7	LP Enabled	<p>Decode of SMC_PMCTRL[STOPM] field to indicate that VLPS, LLS, or VLLSx are the selected power mode the next time the ARM Core enters Deep Sleep.</p> <p>0 Low Power Stop Mode is not enabled.</p> <p>1 Low Power Stop Mode is enabled.</p> <p>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.</p>
8	Very Low Power Mode	<p>Indicates current power mode is VLPx. This bit is not 'sticky' and should always represent whether VLPx is enabled or not.</p> <p>This bit is used to throttle SWD_CLK frequency up/down.</p>
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.

*Table continues on the next page...*

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

Bit	Name	Description
		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	<p>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.</p> <p>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.</p>
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	<p>SLEEPING==1 and SLEEPDEEP==0 indicates wait or VLPW mode.</p> <p>SLEEPING==1 and SLEEPDEEP==1 indicates stop or VLPS mode.</p>
19 – 31	Reserved for future use	Always read 0.

## 9.4 Debug resets

The debug system receives the following sources of reset:

- System POR reset

Conversely, the debug system is capable of generating 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 field in the NVIC Application Interrupt and Reset control register
- A system reset in the DAP control register which allows the debugger to hold the core in reset.

## 9.5 Micro Trace Buffer (MTB)

The Micro Trace Buffer (MTB) provides a simple execution trace capability for the Cortex-M0+ processor.



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

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

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

## 9.6 Debug in low-power modes

In low-power modes, in which the debug modules are kept static or powered off, the debugger cannot gather any debug data for the duration of the low-power mode.

- In the case that the debugger is held static, the debug port returns to full functionality as soon as the low-power mode exits and the system returns to a state with active debug.
- In the case that the debugger logic is powered off, the debugger is reset on recovery and must be reconfigured 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, CPU clk continues to run to support core register access. In these modes in which CPU clk is left active the debug modules have access to core registers but not to system memory resources accessed via the crossbar.

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

### NOTE

With the debug enabled, transitions from Run --> VLPR --> VLPS are still possible.

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

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 to 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.

## 9.7 Debug and security

When flash security is enabled, 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 status register and can determine the current security state of the device. In the case of a secure device, the debugger has the capability of only performing a mass erase operation.

# Chapter 10

## Pinouts and Packaging

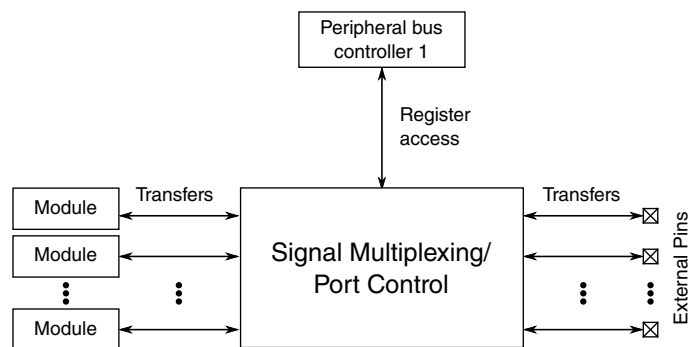
### 10.1 Introduction

To optimize functionality in small packages, pins have several functions available via signal multiplexing. Information found here 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. Refer to that chapter to find which register controls the operation of a specific pin.

### 10.2 Signal multiplexing integration

Information found here 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 I/O Port control and interrupt module features

- 32-bit ports

### NOTE

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

## 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 SIM\_SCGC5[PORTx] to enable the clock. Before turning off the clock, make sure to disable the module. For more details, see 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 KL13 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 32 QFN, 48 QFN, and 64 MAPBGA packages for this product are not yet available. However, these packages are included in Package Your Way program for Kinetis MCUs. Visit [nxp.com/KPYW](http://nxp.com/KPYW) for more details.

80 LQFP	64 LQFP	48 QFN	64 MAP BGA	32 QFN	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
—	—	—	C5	—	NC	NC	NC							
1	1	—	A1	1	PTE0	DISABLED		PTE0/ CLKOUT32K	SPI1_MISO	LPUART1_ TX	RTC_ CLKOUT	CMP0_OUT	I2C1_SDA	
2	2	—	B1	2	PTE1	DISABLED		PTE1	SPI1_MOSI	LPUART1_ RX		SPI1_MISO	I2C1_SCL	
3	—	—	—	—	PTE2	DISABLED		PTE2	SPI1_SCK					
4	—	—	—	—	PTE3	DISABLED		PTE3	SPI1_MISO			SPI1_MOSI		
5	—	—	—	—	PTE4	DISABLED		PTE4	SPI1_PCS0					
6	—	—	—	—	PTE5	DISABLED		PTE5						
7	3	1	—	—	VDD	VDD	VDD							
8	4	2	C4	—	VSS	VSS	VSS							
9	5	3	E1	3	PTE16	ADC0_DP1/ ADC0_SE1	ADC0_DP1/ ADC0_SE1	PTE16	SPIO_PCS0	UART2_TX	TPM_ CLKIN0		FXIO0_D0	
10	6	4	D1	4	PTE17	ADC0_DM1/ ADC0_SE5a	ADC0_DM1/ ADC0_SE5a	PTE17	SPIO_SCK	UART2_RX	TPM_ CLKIN1	LPTMR0_ ALT3	FXIO0_D1	
11	7	5	E2	5	PTE18	ADC0_DP2/ ADC0_SE2	ADC0_DP2/ ADC0_SE2	PTE18	SPIO_MOSI		I2C0_SDA	SPIO_MISO	FXIO0_D2	
12	8	6	D2	6	PTE19	ADC0_DM2/ ADC0_SE6a	ADC0_DM2/ ADC0_SE6a	PTE19	SPIO_MISO		I2C0_SCL	SPIO_MOSI	FXIO0_D3	
13	9	7	G1	—	PTE20	ADC0_DP0/ ADC0_SE0	ADC0_DP0/ ADC0_SE0	PTE20		TPM1_CH0	LPUART0_ TX		FXIO0_D4	
14	10	8	F1	—	PTE21	ADC0_DM0/ ADC0_SE4a	ADC0_DM0/ ADC0_SE4a	PTE21		TPM1_CH1	LPUART0_ RX		FXIO0_D5	
15	11	—	G2	—	PTE22	ADC0_DP3/ ADC0_SE3	ADC0_DP3/ ADC0_SE3	PTE22		TPM2_CH0	UART2_TX		FXIO0_D6	
16	12	—	F2	—	PTE23	ADC0_DM3/ ADC0_SE7a	ADC0_DM3/ ADC0_SE7a	PTE23		TPM2_CH1	UART2_RX		FXIO0_D7	
17	13	9	F4	7	VDDA	VDDA	VDDA							
18	14	10	G4	—	VREFH	VREFH	VREFH							
18	14	10	G4	—	VREFO	VREFO (1,2V reference, bond to VREFH)	VREFH							
19	15	11	G3	—	VREFL	VREFL	VREFL							
20	16	12	F3	8	VSSA	VSSA	VSSA							
21	17	13	H1	—	PTE29	CMP0_IN5/ ADC0_SE4b	CMP0_IN5/ ADC0_SE4b	PTE29		TPM0_CH2	TPM_ CLKIN0			

## KL13 Signal Multiplexing and Pin Assignments

80 LQFP	64 LQFP	48 QFN	64 MAP BGA	32 QFN	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
22	18	14	H2	9	PTE30	DAC0_OUT/ ADC0_ SE23/ CMP0_IN4	DAC0_OUT/ ADC0_ SE23/ CMP0_IN4	PTE30		TPM0_CH3	TPM_ CLKIN1	LPUART1_ TX	LPTMR0_ ALT1	
23	19	—	H3	—	PTE31	DISABLED		PTE31		TPM0_CH4				
24	20	15	H4	—	PTE24	DISABLED		PTE24		TPM0_CH0		I2C0_SCL		
25	21	16	H5	—	PTE25	DISABLED		PTE25		TPM0_CH1		I2C0_SDA		
26	22	17	D3	10	PTA0	SWD_CLK		PTA0		TPM0_CH5				SWD_CLK
27	23	18	D4	11	PTA1	DISABLED		PTA1	LPUART0_ RX	TPM2_CH0				
28	24	19	E5	12	PTA2	DISABLED		PTA2	LPUART0_ TX	TPM2_CH1				
29	25	20	D5	13	PTA3	SWD_DIO		PTA3	I2C1_SCL	TPM0_CH0				SWD_DIO
30	26	21	G5	14	PTA4	NMI_b		PTA4	I2C1_SDA	TPM0_CH1				NMI_b
31	27	—	F5	—	PTA5	DISABLED		PTA5		TPM0_CH2				
32	28	—	H6	—	PTA12	DISABLED		PTA12		TPM1_CH0				
33	29	—	G6	—	PTA13	DISABLED		PTA13		TPM1_CH1				
34	—	—	—	—	PTA14	DISABLED		PTA14	SPI0_PCS0	LPUART0_ TX				
35	—	—	—	—	PTA15	DISABLED		PTA15	SPI0_SCK	LPUART0_ RX				
36	—	—	—	—	PTA16	DISABLED		PTA16	SPI0_MOSI			SPI0_MISO		
37	—	—	—	—	PTA17	DISABLED		PTA17	SPI0_MISO			SPI0_MISO		
38	30	22	G7	15	VDD	VDD	VDD							
39	31	23	H7	16	VSS	VSS	VSS							
40	32	24	H8	17	PTA18	EXTAL0	EXTAL0	PTA18		LPUART1_ RX	TPM_ CLKIN0			
41	33	25	G8	18	PTA19	XTAL0	XTAL0	PTA19		LPUART1_ TX	TPM_ CLKIN1		LPTMR0_ ALT1	
42	34	26	F8	19	PTA20	RESET_b		PTA20						RESET_b
43	35	27	F7	20	PTB0/ LLWU_P5	ADC0_SE8	ADC0_SE8	PTB0/ LLWU_P5	I2C0_SCL	TPM1_CH0	SPI1_MOSI	SPI1_MISO		
44	36	28	F6	21	PTB1	ADC0_SE9	ADC0_SE9	PTB1	I2C0_SDA	TPM1_CH1	SPI1_MISO	SPI1_MOSI		
45	37	29	E7	—	PTB2	ADC0_SE12	ADC0_SE12	PTB2	I2C0_SCL	TPM2_CH0				
46	38	30	E8	—	PTB3	ADC0_SE13	ADC0_SE13	PTB3	I2C0_SDA	TPM2_CH1				
47	—	—	—	—	PTB8			PTB8	SPI1_PCS0	EXTRG_IN				
48	—	—	—	—	PTB9			PTB9	SPI1_SCK					
49	—	—	—	—	PTB10			PTB10	SPI1_PCS0					
50	—	—	—	—	PTB11			PTB11	SPI1_SCK					
51	39	31	E6	—	PTB16			PTB16	SPI1_MOSI	LPUART0_ RX	TPM_ CLKIN0	SPI1_MISO		
52	40	32	D7	—	PTB17			PTB17	SPI1_MISO	LPUART0_ TX	TPM_ CLKIN1	SPI1_MOSI		
53	41	—	D6	—	PTB18			PTB18		TPM2_CH0				

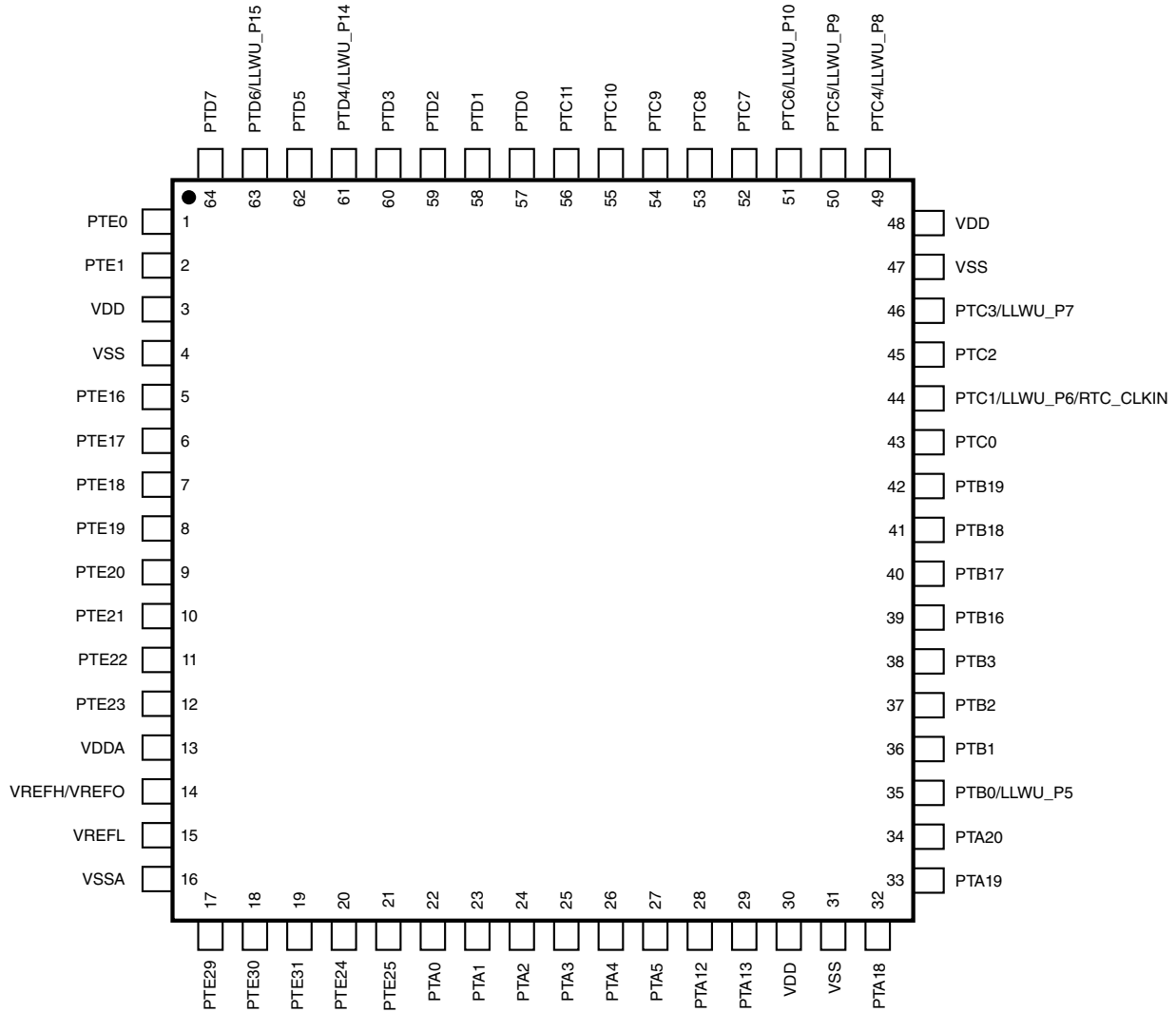
80 LQFP	64 LQFP	48 QFN	64 MAP BGA	32 QFN	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
54	42	—	C7	—	PTB19			PTB19		TPM2_CH1				
55	43	33	D8	—	PTC0	ADC0_SE14	ADC0_SE14	PTC0		EXTRG_IN		CMP0_OUT		
56	44	34	C6	22	PTC1/ LLWU_P6/ RTC_CLKIN	ADC0_SE15	ADC0_SE15	PTC1/ LLWU_P6/ RTC_CLKIN	I2C1_SCL		TPM0_CH0			
57	45	35	B7	23	PTC2	ADC0_SE11	ADC0_SE11	PTC2	I2C1_SDA		TPM0_CH1			
58	46	36	C8	24	PTC3/ LLWU_P7			PTC3/ LLWU_P7	SPI1_SCK	LPUART1_ RX	TPM0_CH2	CLKOUT		
59	47	—	E3	—	VSS	VSS	VSS							
60	48	—	E4	—	VDD	VDD	VDD							
61	49	37	B8	25	PTC4/ LLWU_P8			PTC4/ LLWU_P8	SPI0_PCS0	LPUART1_ TX	TPM0_CH3	SPI1_PCS0		
62	50	38	A8	26	PTC5/ LLWU_P9			PTC5/ LLWU_P9	SPI0_SCK	LPTMR0_ ALT2			CMP0_OUT	
63	51	39	A7	27	PTC6/ LLWU_P10	CMP0_IN0	CMP0_IN0	PTC6/ LLWU_P10	SPI0_MOSI	EXTRG_IN		SPI0_MISO		
64	52	40	B6	28	PTC7	CMP0_IN1	CMP0_IN1	PTC7	SPI0_MISO			SPI0_MISO		
65	53	—	A6	—	PTC8	CMP0_IN2	CMP0_IN2	PTC8	I2C0_SCL	TPM0_CH4				
66	54	—	B5	—	PTC9	CMP0_IN3	CMP0_IN3	PTC9	I2C0_SDA	TPM0_CH5				
67	55	—	B4	—	PTC10			PTC10	I2C1_SCL					
68	56	—	A5	—	PTC11			PTC11	I2C1_SDA					
69	—	—	—	—	PTC12			PTC12			TPM_ CLKIN0			
70	—	—	—	—	PTC13			PTC13			TPM_ CLKIN1			
71	—	—	—	—	PTC16			PTC16						
72	—	—	—	—	PTC17			PTC17						
73	57	41	C3	—	PTD0			PTD0	SPI0_PCS0		TPM0_CH0		FXIO0_D0	
74	58	42	A4	—	PTD1	ADC0_SE5b	ADC0_SE5b	PTD1	SPI0_SCK		TPM0_CH1		FXIO0_D1	
75	59	43	C2	—	PTD2			PTD2	SPI0_MOSI	UART2_RX	TPM0_CH2	SPI0_MISO	FXIO0_D2	
76	60	44	B3	—	PTD3			PTD3	SPI0_MISO	UART2_TX	TPM0_CH3	SPI0_MOSI	FXIO0_D3	
77	61	45	A3	29	PTD4/ LLWU_P14			PTD4/ LLWU_P14	SPI1_PCS0	UART2_RX	TPM0_CH4		FXIO0_D4	
78	62	46	C1	30	PTD5	ADC0_SE6b	ADC0_SE6b	PTD5	SPI1_SCK	UART2_TX	TPM0_CH5		FXIO0_D5	
79	63	47	B2	31	PTD6/ LLWU_P15	ADC0_SE7b	ADC0_SE7b	PTD6/ LLWU_P15	SPI1_MOSI	LPUART0_ RX	I2C1_SDA	SPI1_MISO	FXIO0_D6	
80	64	48	A2	32	PTD7			PTD7	SPI1_MISO	LPUART0_ TX	I2C1_SCL	SPI1_MOSI	FXIO0_D7	

## 10.4 KL13 Family Pinouts

Figure below shows the 64 LQFP pinouts:

**NOTE**

The 32 QFN, 48 QFN, and 64 MAPBGA packages for this product are not yet available. However, these packages are included in Package Your Way program for Kinetis MCUs. Visit [nxp.com/KPYW](http://nxp.com/KPYW) for more details.



**Figure 10-2. 64 LQFP Pinout diagram**

Figure below shows the 80 LQFP pinouts:



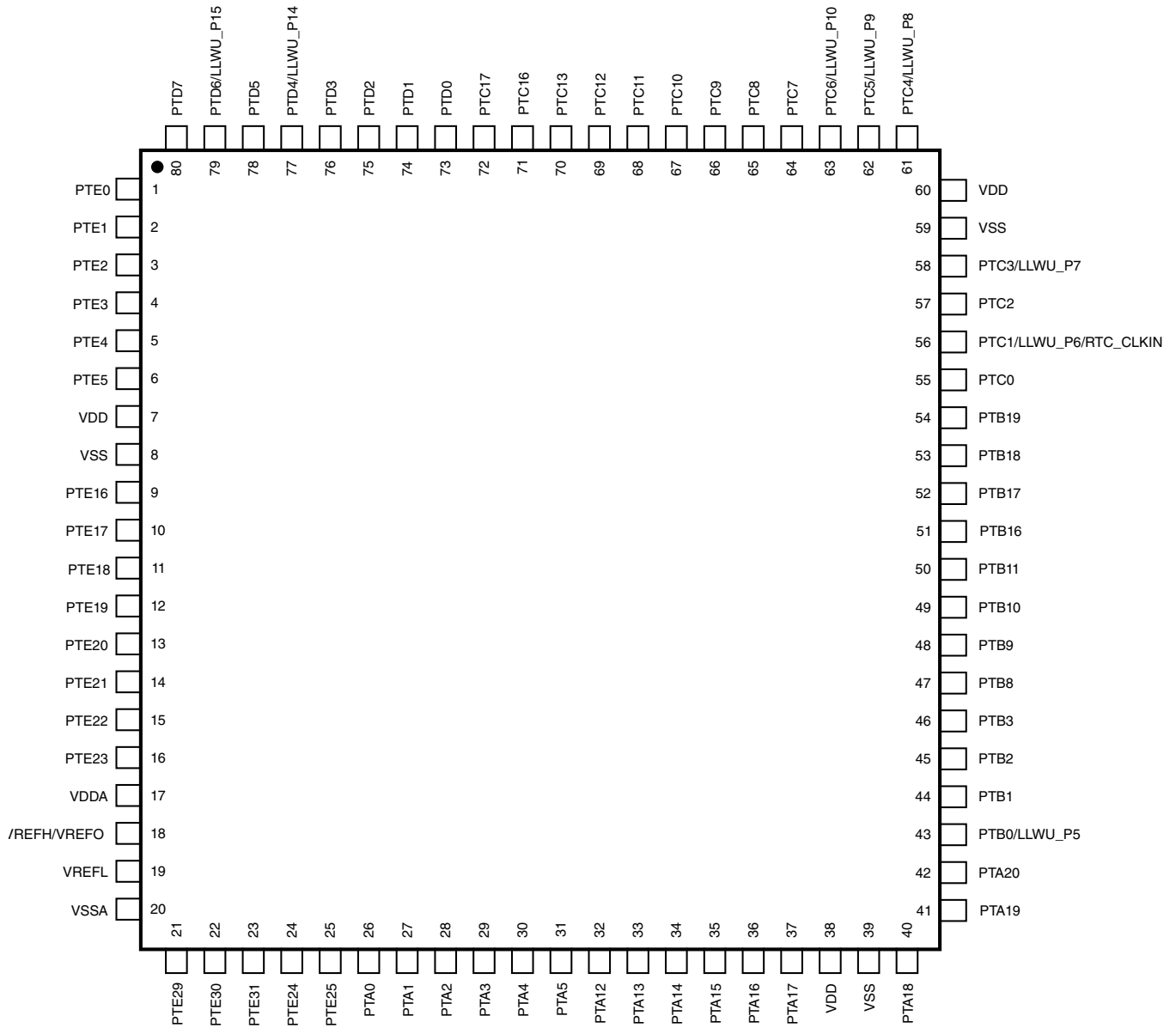


Figure 10-3. 80 LQFP Pinout diagram

Figure below shows the 64 MAPBGA pinouts:

## KL13 Family Pinouts

	1	2	3	4	5	6	7	8	
A	PTE0	PTD7	PTD4/ LLWU_P14	PTD1	PTC11	PTC8	PTC6/ LLWU_P10	PTC5/ LLWU_P9	A
B	PTE1	PTD6/ LLWU_P15	PTD3	PTC10	PTC9	PTC7	PTC2	PTC4/ LLWU_P8	B
C	PTD5	PTD2	PTD0	VSS	NC	PTC1/ LLWU_P6/ RTC_CLKIN	PTB19	PTC3/ LLWU_P7	C
D	PTE17	PTE19	PTA0	PTA1	PTA3	PTB18	PTB17	PTC0	D
E	PTE16	PTE18	VSS	VDD	PTA2	PTB16	PTB2	PTB3	E
F	PTE21	PTE23	VSSA	VDDA	PTA5	PTB1	PTB0/ LLWU_P5	PTA20	F
G	PTE20	PTE22	VREFL	VREFH/ VREFO	PTA4	PTA13	VDD	PTA19	G
H	PTE29	PTE30	PTE31	PTE24	PTE25	PTA12	VSS	PTA18	H
	1	2	3	4	5	6	7	8	

**Figure 10-4. 64 MAPBGA Pinout diagram**

Figure below shows the 48 QFN pinouts:

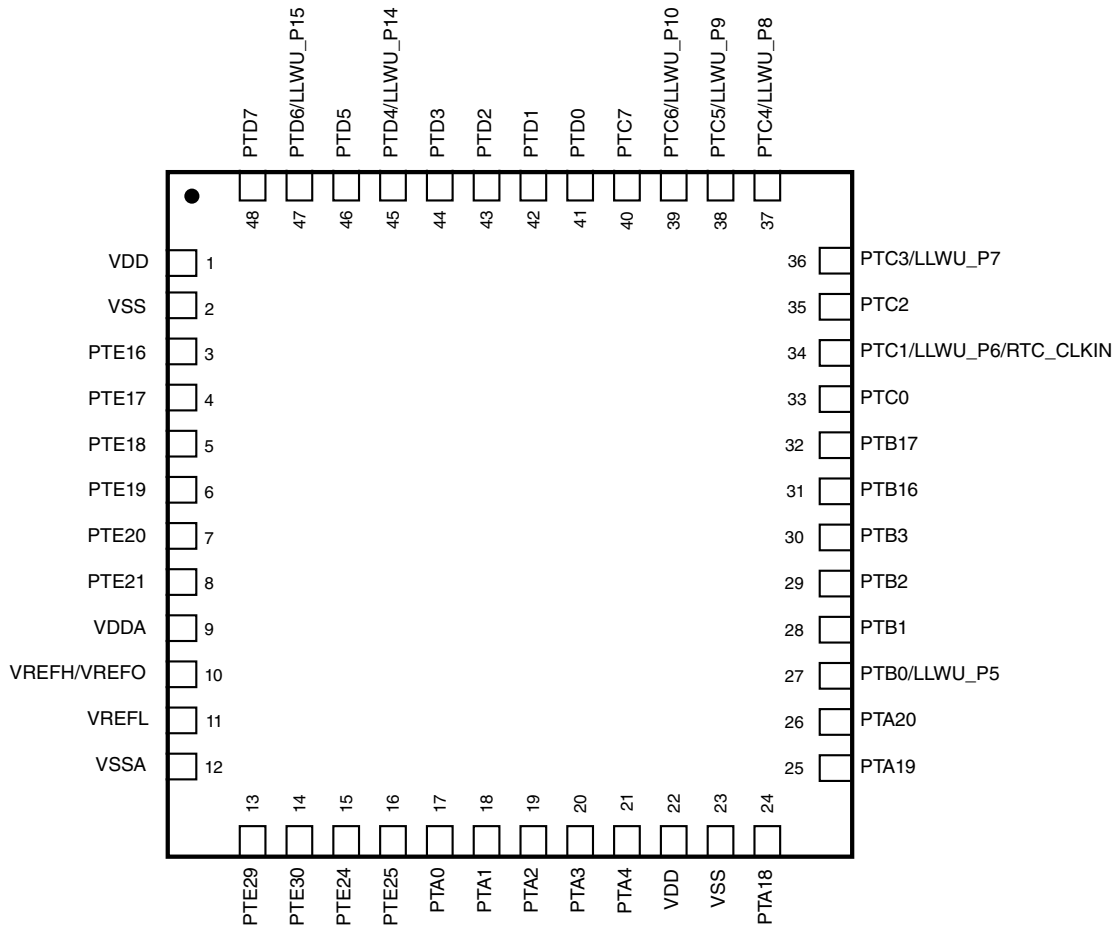


Figure 10-5. 48 QFN Pinout diagram

Figure below shows the 32 QFN pinouts:

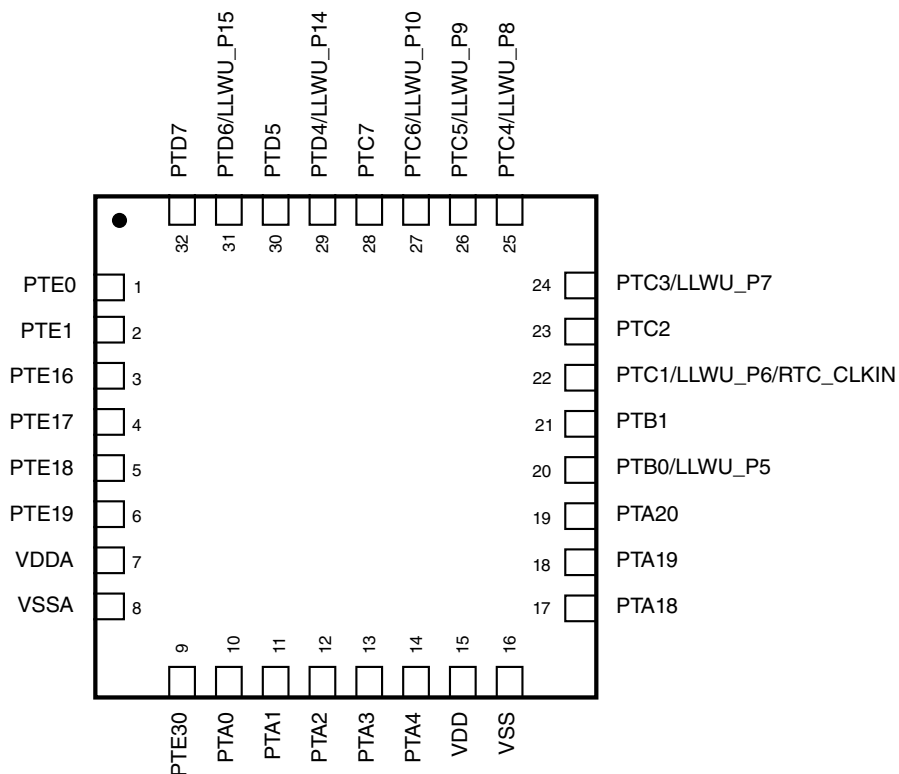


Figure 10-6. 32 QFN Pinout diagram

## 10.5 Module Signal Description Tables

### 10.5.1 Core modules

Table 10-2. SWD signal descriptions

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

## 10.5.2 System modules

**Table 10-3. System signal descriptions**

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

**Table 10-4. LLWU signal descriptions**

Chip signal name	Module signal name	Description	I/O
LLWU_Pn	LLWU_Pn	Wakeup inputs (n = 5, 6, 7, 8, 9, 10, 14, 15)	I

## 10.5.3 Clock modules

**Table 10-5. OSC signal descriptions**

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

## 10.5.4 Memories and memory interfaces

## 10.5.5 Analog

This table presents the signal descriptions of the ADC0 module.

**Table 10-6. ADC0 signal descriptions**

Chip signal name	Module signal name	Description	I/O
ADC0_DPn	DADP3–DADP0	Differential Analog Channel Inputs	I
ADC0_DMn	DADM3–DADM0	Differential Analog Channel Inputs	I

*Table continues on the next page...*

**Table 10-6. ADC0 signal descriptions (continued)**

Chip signal name	Module signal name	Description	I/O
ADC0_SEn	ADn	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
EXTRG_IN	ADHWT	Hardware trigger	I

This table presents the signal descriptions of the CMP0 module.

**Table 10-7. CMP0 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

This table presents the signal descriptions of the DAC0 module.

**Table 10-8. DAC0 signal descriptions**

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

**Table 10-9. VREF signal descriptions**

Chip signal name	Module signal name	Description	I/O
VREF_OUT	VREF_OUT	Internally-generated voltage reference output	O

## 10.5.6 Timer Modules

**Table 10-10. TPM0 signal descriptions**

Chip signal name	Module signal name	Description	I/O
TPM_CLKIN[1:0]	TPM_EXTCLK	External clock. TPM external clock can be selected to increment the TPM counter on every rising edge synchronized to the counter clock.	I

*Table continues on the next page...*

**Table 10-10. TPM0 signal descriptions (continued)**

Chip signal name	Module signal name	Description	I/O
TPM0_CH[5:0]	TPM_CHn	TPM channel (n = 5 to 0). A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input.	I/O

**Table 10-11. TPM1 signal descriptions**

Chip signal name	Module signal name	Description	I/O
TPM_CLKIN[1:0]	TPM_EXTCLK	External clock. TPM external clock can be selected to increment the TPM counter on every rising edge synchronized to the counter clock.	I
TPM1_CH[1:0]	TPM_CHn	TPM channel (n = 1 to 0). A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input.	I/O

**Table 10-12. TPM2 signal descriptions**

Chip signal name	Module signal name	Description	I/O
TPM_CLKIN[1:0]	TPM_EXTCLK	External clock. TPM external clock can be selected to increment the TPM counter on every rising edge synchronized to the counter clock.	I
TPM2_CH[1:0]	TPM_CHn	TPM channel (n = 1 to 0). A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input.	I/O

**Table 10-13. LPTMR0 signal descriptions**

Chip signal name	Module signal name	Description	I/O
LPTMR0_ALT[3:1]	LPTMR0_ALTn	Pulse Counter Input pin	I

**Table 10-14. RTC signal descriptions**

Chip signal name	Module signal name	Description	I/O
RTC_CLKOUT <sup>1</sup>	RTC_CLKOUT	1 Hz square-wave output or OSCERCLK	O

1. RTC\_CLKOUT can also be driven with OSCERCLK via SIM control bit SIM\_SOPT[RCTCLKOUTSEL]

## 10.5.7 Communication interfaces

**Table 10-15. SPI0 signal descriptions**

Chip signal name	Module signal name	Description	I/O
SPI0_MISO	MISO	Master Data In, Slave Data Out	I/O
SPI0_MOSI	MOSI	Master Data Out, Slave Data In	I/O
SPI0_SCLK	SPSCK	SPI Serial Clock	I/O
SPI0_PCS0	$\overline{SS}$	Slave Select	I/O

**Table 10-16. SPI1 signal descriptions**

Chip signal name	Module signal name	Description	I/O
SPI1_MISO	MISO	Master Data In, Slave Data Out	I/O
SPI1_MOSI	MOSI	Master Data Out, Slave Data In	I/O
SPI1_SCLK	SPSCK	SPI Serial Clock	I/O
SPI1_PCS0	$\overline{SS}$	Slave Select	I/O

**Table 10-17. I<sup>2</sup>C0 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 10-18. I<sup>2</sup>C1 signal descriptions**

Chip signal name	Module signal name	Description	I/O
I2C1_SCL	SCL	Bidirectional serial clock line of the I <sup>2</sup> C system.	I/O
I2C1_SDA	SDA	Bidirectional serial data line of the I <sup>2</sup> C system.	I/O

**Table 10-19. LPUART0 signal descriptions**

Chip signal name	Module signal name	Description	I/O
LPUART0_TX	TxD	Transmit data	I/O
LPUART0_RX	RxD	Receive data	I



**Table 10-20. LPUART1 signal descriptions**

Chip signal name	Module signal name	Description	I/O
LPUART1_TX	TxD	Transmit data	I/O
LPUART1_RX	RxD	Receive data	I

**Table 10-21. UART2 signal descriptions**

Chip signal name	Module signal name	Description	I/O
UART2_TX	TxD	Transmit data	O
UART2_RX	RxD	Receive data	I

**Table 10-22. FlexIO signal descriptions**

Chip signal name	Module signal name	Description	I/O
FXIO0_Dx	FXIO_Dn (n=0...7)	Bidirectional FlexIO Shifter and Timer pin inputs/outputs	I/O

## 10.5.8 Human-machine interfaces (HMI)

**Table 10-23. GPIO Signal Descriptions**

Chip signal name	Module signal name	Description	I/O
PTA[20:0]	PORTA20–PORTA0	General-purpose input/output	I/O
PTB[19:0]	PORTB19–PORTB0	General-purpose input/output	I/O
PTC[23:0]	PORTC23–PORTC0	General-purpose input/output	I/O
PTD[7:0]	PORTD7–PORTD0	General-purpose input/output	I/O
PTE[31:0]	PORTE31–PORTE0	General-purpose input/output	I/O



# Chapter 11

## Port Control and Interrupts (PORT)

### 11.1 Chip-specific PORT information

The following table list registers that are not implemented in this device:

Absolute address (hex)	Register names	Notes
4004_9018-4004_902C	PORTA_PCR6, PORTA_PCR7, PORTA_PCR8, PORTA_PCR9, PORTA_PCR10, and PORTA_PCR11	Not implemented in this device.
4004_9054-4004_907C	PORTA_PCR21, PORTA_PCR22, PORTA_PCR23, PORTA_PCR24, PORTA_PCR25, PORTA_PCR26, PORTA_PCR27, PORTA_PCR28, PORTA_PCR29, PORTA_PCR30, and PORTA_PCR31	
4004_A010-4004_A01C	PORTB_PCR4, PORTB_PCR5, PORTB_PCR6, and PORTB_PCR7	
4004_A030-4004_A03C	PORTB_PCR12, PORTB_PCR13, PORTB_PCR14, and PORTB_PCR15	
4004_A050-4004_A07C	PORTB_PCR20, PORTB_PCR21, PORTB_PCR22, PORTB_PCR23, PORTB_PCR24, PORTB_PCR25, PORTB_PCR26, PORTB_PCR27, PORTB_PCR28, PORTB_PCR29, PORTB_PCR30, and PORTB_PCR31	
4004_B038	PORTC_PCR14	
4004_B03C	PORTC_PCR15	
4004_B048	PORTC_PCR18	
4004_B04C	PORTC_PCR19	
4004_B060-4004_B07C	PORTC_PCR24, PORTC_PCR25, PORTC_PCR26, PORTC_PCR27, PORTC_PCR28, PORTC_PCR29, PORTC_PCR30, and PORTC_PCR31	
4004_C020-4004_C07C	PORTD_PCR8, PORTD_PCR9, PORTD_PCR10, PORTD_PCR11, PORTD_PCR12, PORTD_PCR13, PORTD_PCR14, PORTD_PCR15, PORTD_PCR16, PORTD_PCR17,	

*Table continues on the next page...*

## Port control and interrupt summary

Absolute address (hex)	Register names	Notes
	PORTD_PCR18, PORTD_PCR19, PORTD_PCR20, PORTD_PCR21, PORTD_PCR22, PORTD_PCR23, PORTD_PCR24, PORTD_PCR25, PORTD_PCR26, PORTD_PCR27, PORTD_PCR28, PORTD_PCR29, PORTD_PCR30, and PORTD_PCR31	
4004_D018-4004_D03C	PORTE_PCR6, PORTE_PCR7, PORTE_PCR8, PORTE_PCR9, PORTE_PCR10, PORTE_PCR11, PORTE_PCR12, PORTE_PCR13, PORTE_PCR14, and PORTE_PCR15	
4004_D068-4004_D070	PORTE_PCR26, PORTE_PCR27, and PORTE_PCR28	

## 11.2 Port control and interrupt summary

The following table provides more information regarding the Port Control and Interrupt configurations .

**Table 11-1. 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	PTA0=Pull down, Others=Pull up	Pull up	Pull up	Pull up	Pull up
Pull enable control	Yes	Yes	Yes	Yes	Yes
Pull enable at reset	PTA0/PTA3/PTA4/ RESET_b=Enabled ; Others=Disabled	Disabled	Disabled	Disabled	Disabled
Slew rate enable control	Yes	Yes	Yes	Yes	Yes
Slew rate enable at reset	PTA3/PTA14/ PTA15/PTA16/ PTA17=Fast Slew; Others=Slow Slew	PTB0,PTB1,PTB8, PTB9,PTB10,PTB1 1,PTB16,PTB17 = Fast Slew; Others=Slow Slew	PTC3/PTC4/PTC5/ PTC6/PTC7=Fast Slew; Others=Slow Slew	PTD0~7=Fast Slew; Others=Slow Slew	PTE0,PTE1,PTE2, PTE3,PTE4,PTE16 ,PTE17,PTE18,PT E19 =Fast Slew; Others=Slow Slew
Passive filter enable control	PTA4 and RESET_b only	No	No	No	No
Passive filter enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Open drain enable control <sup>1</sup>	No	No	No	No	No
Open drain enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled

*Table continues on the next page...*

**Table 11-1. Ports summary (continued)**

Feature	Port A	Port B	Port C	Port D	Port E
Drive strength enable control	No	PTB0/PTB1 only	PTC3	PTD7 only	No
Drive strength enable at reset	Disabled	Disabled	Disabled	Disabled	Disabled
Pin mux control	Yes	Yes	Yes	Yes	Yes
Pin mux at reset	PTA0/PTA3/ PTA4=ALT7; Others=ALT0	ALT0	ALT0	ALT0	ALT0
Lock bit	No	No	No	No	No
Interrupt and DMA request	Yes	Yes	Yes	Yes	Yes
Digital glitch filter	No	No	No	No	No

1. UART signals can be configured for open-drain using SIM\_SOPT5 register. IIC signals are automatically enabled for open drain when selected.

### NOTE

PTA20 RESET\_b's PUE/PUS are not controlled by PORTA\_PCR20's PUE/PUS, but they are tied to pull up enabled; LPTMR0\_ALT1/2/3 's PUE/PUS is tied to disabled; LCD\_P0/1/2/3/..59/60 's PUE/PUS is tied to disabled.

## 11.3 Introduction

## 11.4 Overview

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

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

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

### 11.4.1 Features

The PORT module has the following features:

- Pin interrupt on selected pins
  - Interrupt flag and enable registers for each pin

- Support for edge sensitive (rising, falling, both) or level sensitive (low, high) configured per pin
- Support for interrupt or DMA request configured per pin
- Asynchronous wake-up in low-power modes
- Pin interrupt is functional in all digital pin muxing modes
- Port control
  - Individual pull control fields with pullup, pulldown, and pull-disable support on selected pins
  - Individual drive strength field supporting high and low drive strength on selected pins
  - Individual slew rate field supporting fast and slow slew rates on selected pins
  - Individual input passive filter field supporting enable and disable of the individual input passive filter on selected pins
  - Individual mux control field supporting analog or pin disabled, GPIO, and up to chip-specific digital functions
  - Pad configuration fields are functional in all digital pin muxing modes.

## **11.4.2 Modes of operation**

### **11.4.2.1 Run mode**

In Run mode, the PORT operates normally.

### **11.4.2.2 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.

### **11.4.2.3 Stop mode**

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

### **11.4.2.4 Debug mode**

In Debug mode, PORT operates normally.

## 11.5 External signal description

The table found here describes the PORT external signal.

**Table 11-2. 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.

## 11.6 Detailed signal description

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

**Table 11-3. 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.

## 11.7 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	<a href="#">See section</a>	<a href="#">11.7.1/133</a>
4004_9004	Pin Control Register n (PORTA_PCR1)	32	R/W	<a href="#">See section</a>	<a href="#">11.7.1/133</a>
4004_9008	Pin Control Register n (PORTA_PCR2)	32	R/W	<a href="#">See section</a>	<a href="#">11.7.1/133</a>

*Table continues on the next page...*

## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_900C	Pin Control Register n (PORTA_PCR3)	32	R/W	See section	11.7.1/133
4004_9010	Pin Control Register n (PORTA_PCR4)	32	R/W	See section	11.7.1/133
4004_9014	Pin Control Register n (PORTA_PCR5)	32	R/W	See section	11.7.1/133
4004_9018	Pin Control Register n (PORTA_PCR6)	32	R/W	See section	11.7.1/133
4004_901C	Pin Control Register n (PORTA_PCR7)	32	R/W	See section	11.7.1/133
4004_9020	Pin Control Register n (PORTA_PCR8)	32	R/W	See section	11.7.1/133
4004_9024	Pin Control Register n (PORTA_PCR9)	32	R/W	See section	11.7.1/133
4004_9028	Pin Control Register n (PORTA_PCR10)	32	R/W	See section	11.7.1/133
4004_902C	Pin Control Register n (PORTA_PCR11)	32	R/W	See section	11.7.1/133
4004_9030	Pin Control Register n (PORTA_PCR12)	32	R/W	See section	11.7.1/133
4004_9034	Pin Control Register n (PORTA_PCR13)	32	R/W	See section	11.7.1/133
4004_9038	Pin Control Register n (PORTA_PCR14)	32	R/W	See section	11.7.1/133
4004_903C	Pin Control Register n (PORTA_PCR15)	32	R/W	See section	11.7.1/133
4004_9040	Pin Control Register n (PORTA_PCR16)	32	R/W	See section	11.7.1/133
4004_9044	Pin Control Register n (PORTA_PCR17)	32	R/W	See section	11.7.1/133
4004_9048	Pin Control Register n (PORTA_PCR18)	32	R/W	See section	11.7.1/133
4004_904C	Pin Control Register n (PORTA_PCR19)	32	R/W	See section	11.7.1/133
4004_9050	Pin Control Register n (PORTA_PCR20)	32	R/W	See section	11.7.1/133
4004_9054	Pin Control Register n (PORTA_PCR21)	32	R/W	See section	11.7.1/133
4004_9058	Pin Control Register n (PORTA_PCR22)	32	R/W	See section	11.7.1/133
4004_905C	Pin Control Register n (PORTA_PCR23)	32	R/W	See section	11.7.1/133
4004_9060	Pin Control Register n (PORTA_PCR24)	32	R/W	See section	11.7.1/133
4004_9064	Pin Control Register n (PORTA_PCR25)	32	R/W	See section	11.7.1/133
4004_9068	Pin Control Register n (PORTA_PCR26)	32	R/W	See section	11.7.1/133
4004_906C	Pin Control Register n (PORTA_PCR27)	32	R/W	See section	11.7.1/133
4004_9070	Pin Control Register n (PORTA_PCR28)	32	R/W	See section	11.7.1/133
4004_9074	Pin Control Register n (PORTA_PCR29)	32	R/W	See section	11.7.1/133
4004_9078	Pin Control Register n (PORTA_PCR30)	32	R/W	See section	11.7.1/133
4004_907C	Pin Control Register n (PORTA_PCR31)	32	R/W	See section	11.7.1/133
4004_9080	Global Pin Control Low Register (PORTA_GPCLR)	32	W (always reads 0)	0000_0000h	11.7.2/136
4004_9084	Global Pin Control High Register (PORTA_GPCHR)	32	W (always reads 0)	0000_0000h	11.7.3/136
4004_90A0	Interrupt Status Flag Register (PORTA_ISFR)	32	w1c	0000_0000h	11.7.4/137
4004_A000	Pin Control Register n (PORTB_PCR0)	32	R/W	See section	11.7.1/133
4004_A004	Pin Control Register n (PORTB_PCR1)	32	R/W	See section	11.7.1/133
4004_A008	Pin Control Register n (PORTB_PCR2)	32	R/W	See section	11.7.1/133

Table continues on the next page...



## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_A00C	Pin Control Register n (PORTB_PCR3)	32	R/W	See section	11.7.1/133
4004_A010	Pin Control Register n (PORTB_PCR4)	32	R/W	See section	11.7.1/133
4004_A014	Pin Control Register n (PORTB_PCR5)	32	R/W	See section	11.7.1/133
4004_A018	Pin Control Register n (PORTB_PCR6)	32	R/W	See section	11.7.1/133
4004_A01C	Pin Control Register n (PORTB_PCR7)	32	R/W	See section	11.7.1/133
4004_A020	Pin Control Register n (PORTB_PCR8)	32	R/W	See section	11.7.1/133
4004_A024	Pin Control Register n (PORTB_PCR9)	32	R/W	See section	11.7.1/133
4004_A028	Pin Control Register n (PORTB_PCR10)	32	R/W	See section	11.7.1/133
4004_A02C	Pin Control Register n (PORTB_PCR11)	32	R/W	See section	11.7.1/133
4004_A030	Pin Control Register n (PORTB_PCR12)	32	R/W	See section	11.7.1/133
4004_A034	Pin Control Register n (PORTB_PCR13)	32	R/W	See section	11.7.1/133
4004_A038	Pin Control Register n (PORTB_PCR14)	32	R/W	See section	11.7.1/133
4004_A03C	Pin Control Register n (PORTB_PCR15)	32	R/W	See section	11.7.1/133
4004_A040	Pin Control Register n (PORTB_PCR16)	32	R/W	See section	11.7.1/133
4004_A044	Pin Control Register n (PORTB_PCR17)	32	R/W	See section	11.7.1/133
4004_A048	Pin Control Register n (PORTB_PCR18)	32	R/W	See section	11.7.1/133
4004_A04C	Pin Control Register n (PORTB_PCR19)	32	R/W	See section	11.7.1/133
4004_A050	Pin Control Register n (PORTB_PCR20)	32	R/W	See section	11.7.1/133
4004_A054	Pin Control Register n (PORTB_PCR21)	32	R/W	See section	11.7.1/133
4004_A058	Pin Control Register n (PORTB_PCR22)	32	R/W	See section	11.7.1/133
4004_A05C	Pin Control Register n (PORTB_PCR23)	32	R/W	See section	11.7.1/133
4004_A060	Pin Control Register n (PORTB_PCR24)	32	R/W	See section	11.7.1/133
4004_A064	Pin Control Register n (PORTB_PCR25)	32	R/W	See section	11.7.1/133
4004_A068	Pin Control Register n (PORTB_PCR26)	32	R/W	See section	11.7.1/133
4004_A06C	Pin Control Register n (PORTB_PCR27)	32	R/W	See section	11.7.1/133
4004_A070	Pin Control Register n (PORTB_PCR28)	32	R/W	See section	11.7.1/133
4004_A074	Pin Control Register n (PORTB_PCR29)	32	R/W	See section	11.7.1/133
4004_A078	Pin Control Register n (PORTB_PCR30)	32	R/W	See section	11.7.1/133
4004_A07C	Pin Control Register n (PORTB_PCR31)	32	R/W	See section	11.7.1/133
4004_A080	Global Pin Control Low Register (PORTB_GPCLR)	32	W (always reads 0)	0000_0000h	11.7.2/136
4004_A084	Global Pin Control High Register (PORTB_GPCHR)	32	W (always reads 0)	0000_0000h	11.7.3/136
4004_A0A0	Interrupt Status Flag Register (PORTB_ISFR)	32	w1c	0000_0000h	11.7.4/137
4004_B000	Pin Control Register n (PORTC_PCR0)	32	R/W	See section	11.7.1/133
4004_B004	Pin Control Register n (PORTC_PCR1)	32	R/W	See section	11.7.1/133
4004_B008	Pin Control Register n (PORTC_PCR2)	32	R/W	See section	11.7.1/133

Table continues on the next page...

## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_B00C	Pin Control Register n (PORTC_PCR3)	32	R/W	See section	11.7.1/133
4004_B010	Pin Control Register n (PORTC_PCR4)	32	R/W	See section	11.7.1/133
4004_B014	Pin Control Register n (PORTC_PCR5)	32	R/W	See section	11.7.1/133
4004_B018	Pin Control Register n (PORTC_PCR6)	32	R/W	See section	11.7.1/133
4004_B01C	Pin Control Register n (PORTC_PCR7)	32	R/W	See section	11.7.1/133
4004_B020	Pin Control Register n (PORTC_PCR8)	32	R/W	See section	11.7.1/133
4004_B024	Pin Control Register n (PORTC_PCR9)	32	R/W	See section	11.7.1/133
4004_B028	Pin Control Register n (PORTC_PCR10)	32	R/W	See section	11.7.1/133
4004_B02C	Pin Control Register n (PORTC_PCR11)	32	R/W	See section	11.7.1/133
4004_B030	Pin Control Register n (PORTC_PCR12)	32	R/W	See section	11.7.1/133
4004_B034	Pin Control Register n (PORTC_PCR13)	32	R/W	See section	11.7.1/133
4004_B038	Pin Control Register n (PORTC_PCR14)	32	R/W	See section	11.7.1/133
4004_B03C	Pin Control Register n (PORTC_PCR15)	32	R/W	See section	11.7.1/133
4004_B040	Pin Control Register n (PORTC_PCR16)	32	R/W	See section	11.7.1/133
4004_B044	Pin Control Register n (PORTC_PCR17)	32	R/W	See section	11.7.1/133
4004_B048	Pin Control Register n (PORTC_PCR18)	32	R/W	See section	11.7.1/133
4004_B04C	Pin Control Register n (PORTC_PCR19)	32	R/W	See section	11.7.1/133
4004_B050	Pin Control Register n (PORTC_PCR20)	32	R/W	See section	11.7.1/133
4004_B054	Pin Control Register n (PORTC_PCR21)	32	R/W	See section	11.7.1/133
4004_B058	Pin Control Register n (PORTC_PCR22)	32	R/W	See section	11.7.1/133
4004_B05C	Pin Control Register n (PORTC_PCR23)	32	R/W	See section	11.7.1/133
4004_B060	Pin Control Register n (PORTC_PCR24)	32	R/W	See section	11.7.1/133
4004_B064	Pin Control Register n (PORTC_PCR25)	32	R/W	See section	11.7.1/133
4004_B068	Pin Control Register n (PORTC_PCR26)	32	R/W	See section	11.7.1/133
4004_B06C	Pin Control Register n (PORTC_PCR27)	32	R/W	See section	11.7.1/133
4004_B070	Pin Control Register n (PORTC_PCR28)	32	R/W	See section	11.7.1/133
4004_B074	Pin Control Register n (PORTC_PCR29)	32	R/W	See section	11.7.1/133
4004_B078	Pin Control Register n (PORTC_PCR30)	32	R/W	See section	11.7.1/133
4004_B07C	Pin Control Register n (PORTC_PCR31)	32	R/W	See section	11.7.1/133
4004_B080	Global Pin Control Low Register (PORTC_GPCLR)	32	W (always reads 0)	0000_0000h	11.7.2/136
4004_B084	Global Pin Control High Register (PORTC_GPCHR)	32	W (always reads 0)	0000_0000h	11.7.3/136
4004_B0A0	Interrupt Status Flag Register (PORTC_ISFR)	32	w1c	0000_0000h	11.7.4/137
4004_C000	Pin Control Register n (PORTD_PCR0)	32	R/W	See section	11.7.1/133
4004_C004	Pin Control Register n (PORTD_PCR1)	32	R/W	See section	11.7.1/133
4004_C008	Pin Control Register n (PORTD_PCR2)	32	R/W	See section	11.7.1/133

Table continues on the next page...

## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_C00C	Pin Control Register n (PORTD_PCR3)	32	R/W	See section	11.7.1/133
4004_C010	Pin Control Register n (PORTD_PCR4)	32	R/W	See section	11.7.1/133
4004_C014	Pin Control Register n (PORTD_PCR5)	32	R/W	See section	11.7.1/133
4004_C018	Pin Control Register n (PORTD_PCR6)	32	R/W	See section	11.7.1/133
4004_C01C	Pin Control Register n (PORTD_PCR7)	32	R/W	See section	11.7.1/133
4004_C020	Pin Control Register n (PORTD_PCR8)	32	R/W	See section	11.7.1/133
4004_C024	Pin Control Register n (PORTD_PCR9)	32	R/W	See section	11.7.1/133
4004_C028	Pin Control Register n (PORTD_PCR10)	32	R/W	See section	11.7.1/133
4004_C02C	Pin Control Register n (PORTD_PCR11)	32	R/W	See section	11.7.1/133
4004_C030	Pin Control Register n (PORTD_PCR12)	32	R/W	See section	11.7.1/133
4004_C034	Pin Control Register n (PORTD_PCR13)	32	R/W	See section	11.7.1/133
4004_C038	Pin Control Register n (PORTD_PCR14)	32	R/W	See section	11.7.1/133
4004_C03C	Pin Control Register n (PORTD_PCR15)	32	R/W	See section	11.7.1/133
4004_C040	Pin Control Register n (PORTD_PCR16)	32	R/W	See section	11.7.1/133
4004_C044	Pin Control Register n (PORTD_PCR17)	32	R/W	See section	11.7.1/133
4004_C048	Pin Control Register n (PORTD_PCR18)	32	R/W	See section	11.7.1/133
4004_C04C	Pin Control Register n (PORTD_PCR19)	32	R/W	See section	11.7.1/133
4004_C050	Pin Control Register n (PORTD_PCR20)	32	R/W	See section	11.7.1/133
4004_C054	Pin Control Register n (PORTD_PCR21)	32	R/W	See section	11.7.1/133
4004_C058	Pin Control Register n (PORTD_PCR22)	32	R/W	See section	11.7.1/133
4004_C05C	Pin Control Register n (PORTD_PCR23)	32	R/W	See section	11.7.1/133
4004_C060	Pin Control Register n (PORTD_PCR24)	32	R/W	See section	11.7.1/133
4004_C064	Pin Control Register n (PORTD_PCR25)	32	R/W	See section	11.7.1/133
4004_C068	Pin Control Register n (PORTD_PCR26)	32	R/W	See section	11.7.1/133
4004_C06C	Pin Control Register n (PORTD_PCR27)	32	R/W	See section	11.7.1/133
4004_C070	Pin Control Register n (PORTD_PCR28)	32	R/W	See section	11.7.1/133
4004_C074	Pin Control Register n (PORTD_PCR29)	32	R/W	See section	11.7.1/133
4004_C078	Pin Control Register n (PORTD_PCR30)	32	R/W	See section	11.7.1/133
4004_C07C	Pin Control Register n (PORTD_PCR31)	32	R/W	See section	11.7.1/133
4004_C080	Global Pin Control Low Register (PORTD_GPCLR)	32	W (always reads 0)	0000_0000h	11.7.2/136
4004_C084	Global Pin Control High Register (PORTD_GPCHR)	32	W (always reads 0)	0000_0000h	11.7.3/136
4004_C0A0	Interrupt Status Flag Register (PORTD_ISFR)	32	w1c	0000_0000h	11.7.4/137
4004_D000	Pin Control Register n (PORTE_PCR0)	32	R/W	See section	11.7.1/133
4004_D004	Pin Control Register n (PORTE_PCR1)	32	R/W	See section	11.7.1/133
4004_D008	Pin Control Register n (PORTE_PCR2)	32	R/W	See section	11.7.1/133

Table continues on the next page...

## PORT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4004_D00C	Pin Control Register n (PORTE_PCR3)	32	R/W	See section	11.7.1/133
4004_D010	Pin Control Register n (PORTE_PCR4)	32	R/W	See section	11.7.1/133
4004_D014	Pin Control Register n (PORTE_PCR5)	32	R/W	See section	11.7.1/133
4004_D018	Pin Control Register n (PORTE_PCR6)	32	R/W	See section	11.7.1/133
4004_D01C	Pin Control Register n (PORTE_PCR7)	32	R/W	See section	11.7.1/133
4004_D020	Pin Control Register n (PORTE_PCR8)	32	R/W	See section	11.7.1/133
4004_D024	Pin Control Register n (PORTE_PCR9)	32	R/W	See section	11.7.1/133
4004_D028	Pin Control Register n (PORTE_PCR10)	32	R/W	See section	11.7.1/133
4004_D02C	Pin Control Register n (PORTE_PCR11)	32	R/W	See section	11.7.1/133
4004_D030	Pin Control Register n (PORTE_PCR12)	32	R/W	See section	11.7.1/133
4004_D034	Pin Control Register n (PORTE_PCR13)	32	R/W	See section	11.7.1/133
4004_D038	Pin Control Register n (PORTE_PCR14)	32	R/W	See section	11.7.1/133
4004_D03C	Pin Control Register n (PORTE_PCR15)	32	R/W	See section	11.7.1/133
4004_D040	Pin Control Register n (PORTE_PCR16)	32	R/W	See section	11.7.1/133
4004_D044	Pin Control Register n (PORTE_PCR17)	32	R/W	See section	11.7.1/133
4004_D048	Pin Control Register n (PORTE_PCR18)	32	R/W	See section	11.7.1/133
4004_D04C	Pin Control Register n (PORTE_PCR19)	32	R/W	See section	11.7.1/133
4004_D050	Pin Control Register n (PORTE_PCR20)	32	R/W	See section	11.7.1/133
4004_D054	Pin Control Register n (PORTE_PCR21)	32	R/W	See section	11.7.1/133
4004_D058	Pin Control Register n (PORTE_PCR22)	32	R/W	See section	11.7.1/133
4004_D05C	Pin Control Register n (PORTE_PCR23)	32	R/W	See section	11.7.1/133
4004_D060	Pin Control Register n (PORTE_PCR24)	32	R/W	See section	11.7.1/133
4004_D064	Pin Control Register n (PORTE_PCR25)	32	R/W	See section	11.7.1/133
4004_D068	Pin Control Register n (PORTE_PCR26)	32	R/W	See section	11.7.1/133
4004_D06C	Pin Control Register n (PORTE_PCR27)	32	R/W	See section	11.7.1/133
4004_D070	Pin Control Register n (PORTE_PCR28)	32	R/W	See section	11.7.1/133
4004_D074	Pin Control Register n (PORTE_PCR29)	32	R/W	See section	11.7.1/133
4004_D078	Pin Control Register n (PORTE_PCR30)	32	R/W	See section	11.7.1/133
4004_D07C	Pin Control Register n (PORTE_PCR31)	32	R/W	See section	11.7.1/133
4004_D080	Global Pin Control Low Register (PORTE_GPCLR)	32	W (always reads 0)	0000_0000h	11.7.2/136
4004_D084	Global Pin Control High Register (PORTE_GPCHR)	32	W (always reads 0)	0000_0000h	11.7.3/136
4004_D0A0	Interrupt Status Flag Register (PORTE_ISFR)	32	w1c	0000_0000h	11.7.4/137

## 11.7.1 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								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0					MUX			0	DSE	0	PFE	0	SRE	PE	PS
W																
Reset	0	0	0	0	0	*	*	*	0	*	0	*	0	*	*	*

\* Notes:

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

### PORTx\_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  This field is read-only for pins that do not support interrupt generation.  The pin interrupt configuration is valid in all digital pin muxing modes.

*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>This field is read-only for pins that do not support interrupt generation.</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 Status Flag (ISF) is disabled.</p> <p>0001 ISF flag and DMA request on rising edge.</p> <p>0010 ISF flag and DMA request on falling edge.</p> <p>0011 ISF flag and DMA request on either edge.</p> <p>0100 Reserved.</p> <p>0101 Reserved.</p> <p>0110 Reserved.</p> <p>0111 Reserved.</p> <p>1000 ISF flag and Interrupt when logic 0.</p> <p>1001 ISF flag and Interrupt on rising-edge.</p> <p>1010 ISF flag and Interrupt on falling-edge.</p> <p>1011 ISF flag and Interrupt on either edge.</p> <p>1100 ISF flag and Interrupt when logic 1.</p> <p>1101 Reserved.</p> <p>1110 Reserved.</p> <p>1111 Reserved.</p>
15–11 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>

*Table continues on the next page...*

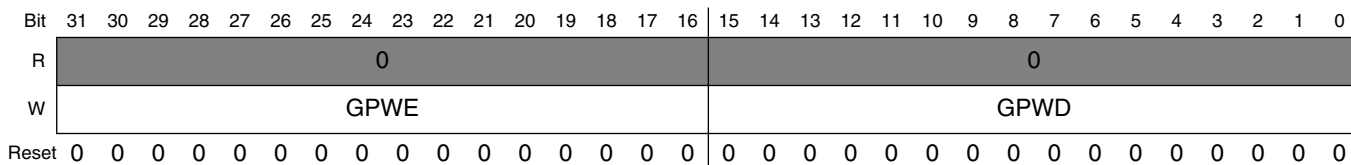
**PORTx\_PCRn field descriptions (continued)**

<b>Field</b>	<b>Description</b>
6 DSE	<p>Drive Strength Enable</p> <p>This field is read-only for pins that do not support a configurable drive strength.</p> <p>Drive strength configuration is valid in all digital pin muxing modes.</p> <p>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.</p>
5 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
4 PFE	<p>Passive Filter Enable</p> <p>This field is read-only for pins that do not support a configurable passive input filter.</p> <p>Passive filter configuration is valid in all digital pin muxing modes.</p> <p>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.</p>
3 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
2 SRE	<p>Slew Rate Enable</p> <p>This field is read-only for pins that do not support a configurable slew rate.</p> <p>Slew rate configuration is valid in all digital pin muxing modes.</p> <p>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.</p>
1 PE	<p>Pull Enable</p> <p>This field is read-only for pins that do not support a configurable pull resistor. Refer to the Chapter of Signal Multiplexing and Signal Descriptions for the pins that support a configurable pull resistor.</p> <p>Pull configuration is valid in all digital pin muxing modes.</p> <p>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.</p>
0 PS	<p>Pull Select</p> <p>This bit is read only for pins that do not support a configurable pull resistor direction.</p> <p>Pull configuration is valid in all digital pin muxing modes.</p> <p>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.</p>

### 11.7.2 Global Pin Control Low Register (PORTx\_GPCLR)

Only 32-bit writes are supported to this register.

Address: Base address + 80h offset



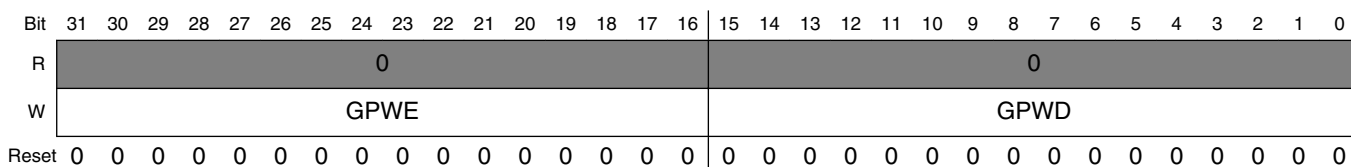
#### PORTx\_GPCLR field descriptions

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.</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>
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>

### 11.7.3 Global Pin Control High Register (PORTx\_GPCHR)

Only 32-bit writes are supported to this register.

Address: Base address + 84h offset



#### 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.</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>
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>



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

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

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

Address: Base address + A0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																	ISF															
W																	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### PORTx\_ISFR field descriptions

Field	Description
ISF	<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>

## 11.8 Functional description

### 11.8.1 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 LK bit (bit 15 of Pin Control Register PCRn) locks the lower 16-bits of each Pin Control register and blocks any writes to that register until the next system reset.

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

## Functional description

- Pullup or pulldown enable on selected pins
- Drive strength and slew rate configuration on selected pins
- Passive input filter enable on selected pins
- Pin Muxing mode

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

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

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

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

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

### 11.8.2 Global pin control

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

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

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

### 11.8.3 External interrupts

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

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

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

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

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

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

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



# Chapter 12

## System Integration Module (SIM)

### 12.1 Chip-specific SIM information

#### 12.1.1 COP clocks

The multiple clock inputs for the COP are:

- 1 kHz clock
- bus clock
- 8 MHz or 2 MHz internal reference clock
- external crystal

### 12.2 Introduction

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

#### 12.2.1 Features

- System clocking configuration
  - System clock divide values
  - Architectural clock gating control
  - ERCLK32K clock selection
  - LPUART and TPM clock selection
- Flash and System RAM size configuration
- TPM external clock and input capture selection
- LPUART receive/transmit source selection/configuration

## 12.3 Memory map and register definition

The SIM module contains many bitfields for selecting the clock source and dividers for various module clocks.

### NOTE

The SIM registers can be written only in supervisor mode. In user mode, write accesses are blocked and will result in a bus error.

### NOTE

The SIM\_SOPT1 is 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	0000_0000h	<a href="#">12.3.1/143</a>
4004_8004	System Options Register 2 (SIM_SOPT2)	32	R/W	0000_0000h	<a href="#">12.3.2/144</a>
4004_800C	System Options Register 4 (SIM_SOPT4)	32	R/W	0000_0000h	<a href="#">12.3.3/146</a>
4004_8010	System Options Register 5 (SIM_SOPT5)	32	R/W	0000_0000h	<a href="#">12.3.4/147</a>
4004_8018	System Options Register 7 (SIM_SOPT7)	32	R/W	0000_0000h	<a href="#">12.3.5/149</a>
4004_8024	System Device Identification Register (SIM_SDID)	32	R	<a href="#">See section</a>	<a href="#">12.3.6/150</a>
4004_8034	System Clock Gating Control Register 4 (SIM_SCGC4)	32	R/W	F000_0030h	<a href="#">12.3.7/151</a>
4004_8038	System Clock Gating Control Register 5 (SIM_SCGC5)	32	R/W	0000_0182h	<a href="#">12.3.8/153</a>
4004_803C	System Clock Gating Control Register 6 (SIM_SCGC6)	32	R/W	0000_0001h	<a href="#">12.3.9/155</a>
4004_8040	System Clock Gating Control Register 7 (SIM_SCGC7)	32	R/W	0000_0100h	<a href="#">12.3.10/157</a>
4004_8044	System Clock Divider Register 1 (SIM_CLKDIV1)	32	R/W	<a href="#">See section</a>	<a href="#">12.3.11/158</a>
4004_804C	Flash Configuration Register 1 (SIM_FCFG1)	32	R/W	<a href="#">See section</a>	<a href="#">12.3.12/159</a>
4004_8050	Flash Configuration Register 2 (SIM_FCFG2)	32	R	<a href="#">See section</a>	<a href="#">12.3.13/161</a>
4004_8058	Unique Identification Register Mid-High (SIM_UIDMH)	32	R	<a href="#">See section</a>	<a href="#">12.3.14/162</a>
4004_805C	Unique Identification Register Mid Low (SIM_UIDML)	32	R	<a href="#">See section</a>	<a href="#">12.3.15/162</a>
4004_8060	Unique Identification Register Low (SIM_UIDL)	32	R	<a href="#">See section</a>	<a href="#">12.3.16/163</a>
4004_8100	COP Control Register (SIM_COPC)	32	R/W	0000_000Ch	<a href="#">12.3.17/163</a>
4004_8104	Service COP (SIM_SRVCOP)	32	W	0000_0000h	<a href="#">12.3.18/165</a>

## 12.3.1 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	0												OSC32KSEL		OSC32KOUT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

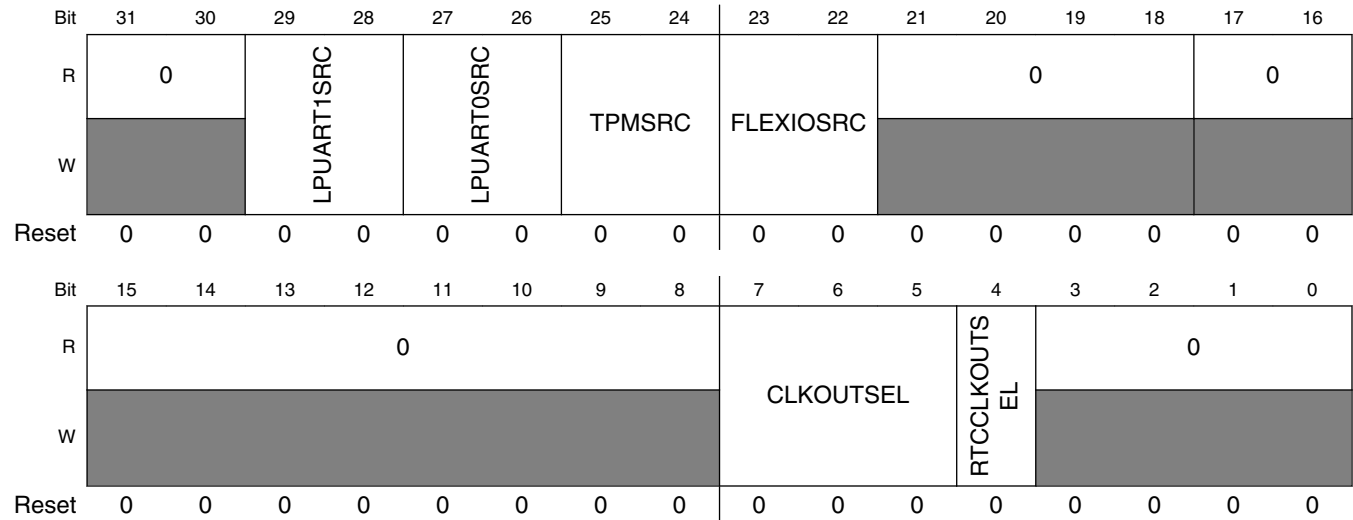
### SIM\_SOPT1 field descriptions

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

### 12.3.2 System Options Register 2 (SIM\_SOPT2)

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

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



#### SIM\_SOPT2 field descriptions

Field	Description
31–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–28 LPUART1SRC	LPUART1 Clock Source Select Selects the clock source for the LPUART1 transmit and receive clock.  00 Clock disabled 01 IRC48M clock 10 OSCERCLK clock 11 MCGIRCLK clock
27–26 LPUART0SRC	LPUART0 Clock Source Select Selects the clock source for the LPUART0 transmit and receive clock.  00 Clock disabled 01 IRC48M clock 10 OSCERCLK clock 11 MCGIRCLK clock
25–24 TPMSRC	TPM Clock Source Select Selects the clock source for the TPM counter clock

Table continues on the next page...



**SIM\_SOPT2 field descriptions (continued)**

Field	Description
	00 Clock disabled 01 IRC48M clock 10 OSCERCLK clock 11 MCGIRCLK clock
23–22 FLEXIOSRC	FlexIO Module Clock Source Select  Selects the clock source for the FlexIO transmit and receive clock.  00 Clock disabled 01 IRC48M clock 10 OSCERCLK clock 11 MCGIRCLK clock
21–18 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.
7–5 CLKOUTSEL	CLKOUT select  Selects the clock to output on the CLKOUT pin.  000 Reserved 001 Reserved 010 Bus clock 011 LPO clock (1 kHz) 100 LIRC_CLK 101 Reserved 110 OSCERCLK 111 IRC48M clock (IRC48M clock can be output to PAD only when chip VDD is 2.7-3.6 V)
4 RTCCLKOUTSEL	RTC Clock Out Select  Selects either the RTC 1 Hz clock or the OSC clock to be output on the RTC_CLKOUT pin.  0 RTC 1 Hz clock is output on the RTC_CLKOUT pin. 1 OSCERCLK clock is output on the RTC_CLKOUT pin.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

### 12.3.3 System Options Register 4 (SIM\_SOPT4)

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0					TPM2CLKSEL	TPM1CLKSEL	TPM0CLKSEL	0			TPM2CH0SRC	0	TPM1CH0SRC	0	
W	[Greyed out]					TPM2CLKSEL	TPM1CLKSEL	TPM0CLKSEL	[Greyed out]			TPM2CH0SRC	[Greyed out]	TPM1CH0SRC	[Greyed out]	
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	[Greyed out]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

#### SIM\_SOPT4 field descriptions

Field	Description
31–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TPM2CLKSEL	TPM2 External Clock Pin Select Selects the external pin used to drive the clock to the TPM2 module. <b>NOTE:</b> The selected pin must also be configured for the TPM external clock function through the appropriate Pin Control Register in the Port Control module. 0 TPM2 external clock driven by TPM_CLKIN0 pin. 1 TPM2 external clock driven by TPM_CLKIN1 pin.
25 TPM1CLKSEL	TPM1 External Clock Pin Select Selects the external pin used to drive the clock to the TPM1 module. <b>NOTE:</b> The selected pin must also be configured for the TPM external clock function through the appropriate pin control register in the port control module. 0 TPM1 external clock driven by TPM_CLKIN0 pin. 1 TPM1 external clock driven by TPM_CLKIN1 pin.
24 TPM0CLKSEL	TPM0 External Clock Pin Select Selects the external pin used to drive the clock to the TPM0 module. <b>NOTE:</b> The selected pin must also be configured for the TPM external clock function through the appropriate pin control register in the port control module. 0 TPM0 external clock driven by TPM_CLKIN0 pin. 1 TPM0 external clock driven by TPM_CLKIN1 pin.

Table continues on the next page...

**SIM\_SOPT4 field descriptions (continued)**

Field	Description
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 TPM2CH0SRC	TPM2 Channel 0 Input Capture Source Select Selects the source for TPM2 channel 0 input capture. <b>NOTE:</b> When TPM2 is not in input capture mode, clear this field. 0 TPM2_CH0 signal 1 CMP0 output
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 TPM1CH0SRC	TPM1 Channel 0 Input Capture Source Select Selects the source for TPM1 channel 0 input capture. <b>NOTE:</b> When TPM1 is not in input capture mode, clear this field. 0 TPM1_CH0 signal 1 CMP0 output
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

**12.3.4 System Options Register 5 (SIM\_SOPT5)**

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

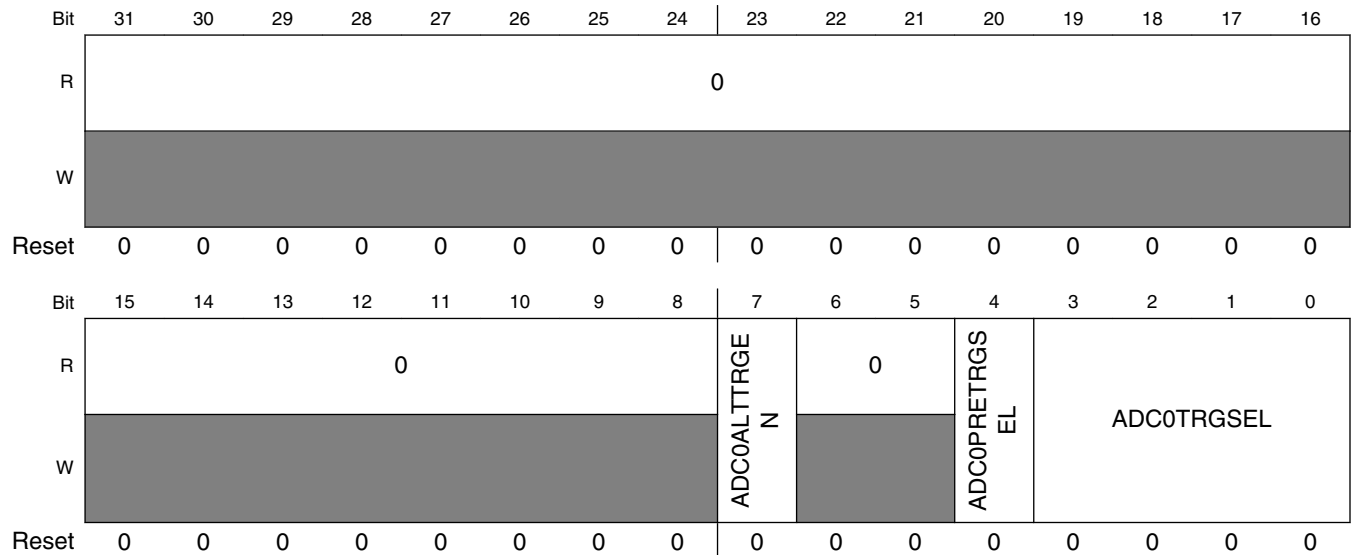
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0													0	UART2ODE	LPUART1ODE	LPUART0ODE
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0								LPUART1RXS RC	LPUART1TXS RC	0		LPUART0RXS RC	LPUART0TXS RC			
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

## SIM\_SOPT5 field descriptions

Field	Description
31–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 UART2ODE	UART2 Open Drain Enable 0 Open drain is disabled on UART2 1 Open drain is enabled on UART2
17 LPUART1ODE	LPUART1 Open Drain Enable 0 Open drain is disabled on LPUART1. 1 Open drain is enabled on LPUART1
16 LPUART0ODE	LPUART0 Open Drain Enable 0 Open drain is disabled on LPUART0. 1 Open drain is enabled on LPUART0.
15–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 LPUART1RXSRC	LPUART1 Receive Data Source Select Selects the source for the LPUART1 receive data. 0 LPUART1_RX pin 1 CMP0 output
5–4 LPUART1TXSRC	LPUART1 Transmit Data Source Select Selects the source for the LPUART1 transmit data. 00 LPUART1_TX pin 01 LPUART1_TX pin modulated with TPM1 channel 0 output 10 LPUART1_TX pin modulated with TPM2 channel 0 output 11 Reserved
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 LPUART0RXSRC	LPUART0 Receive Data Source Select Selects the source for the LPUART0 receive data. 0 LPUART_RX pin 1 CMP0 output
LPUART0TXSRC	LPUART0 Transmit Data Source Select Selects the source for the LPUART0 transmit data. 00 LPUART0_TX pin 01 LPUART0_TX pin modulated with TPM1 channel 0 output 10 LPUART0_TX pin modulated with TPM2 channel 0 output 11 Reserved

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

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



#### SIM\_SOPT7 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADC0ALTTTRGEN	ADC0 Alternate Trigger Enable Enables alternative conversion triggers for ADC0.  0 ADC ADHWT trigger comes from TPM1 channel 0 and channel1. Prior to the assertion of TPM1 channel 0, a pre-trigger pulse will be sent to ADHWTS A to initiate an ADC acquisition using ADCx_SC1A configuration and store ADC conversion in ADCx_RA Register. Prior to the assertion of TPM1 channel 1 a pre-trigger pulse will be sent to ADHWTS B to initiate an ADC acquisition using ADCx_SC1B configuration and store ADC conversion in ADCx_RB Register. 1 ADC ADHWT trigger comes from a peripheral event selected by ADC0TRGSEL bits. ADC0PRETRGSEL bit will select the optional ADHWTS A or ADHWTS B select lines for choosing the ADCx_SC1x config and ADCx_Rx result register to store the ADC conversion.
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. <b>NOTE:</b> The ADC0PRETRGSEL function is ignored if ADC0ALTTTRGEN = 0.  0 Pre-trigger ADHWTS A is selected, thus ADC0 will use ADC0_SC1A configuration for the next ADC conversion and store the result in ADC0_RA register. 1 Pre-trigger ADHWTS B is selected, thus ADC0 will use ADC0_SC1B configuration for the next ADC conversion and store the result in ADC0_RB register.

Table continues on the next page...

**SIM\_SOPT7 field descriptions (continued)**

Field	Description
ADC0TRGSEL	<p>ADC0 Trigger Select</p> <p>Selects 1 of 16 peripherals to initiate an ADC conversion via the ADHWDT input, when ADC0ALTTRGEN =1, else is ignored by ADC0.</p> <p>0000 External trigger pin input (EXTRG_IN)</p> <p>0001 CMP0 output</p> <p>0010 Reserved</p> <p>0011 Reserved</p> <p>0100 PIT trigger 0</p> <p>0101 PIT trigger 1</p> <p>0110 Reserved</p> <p>0111 Reserved</p> <p>1000 TPM0 overflow</p> <p>1001 TPM1 overflow</p> <p>1010 TPM2 overflow</p> <p>1011 Reserved</p> <p>1100 RTC alarm</p> <p>1101 RTC seconds</p> <p>1110 LPTMR0 trigger</p> <p>1111 Reserved</p>

**12.3.6 System Device Identification Register (SIM\_SDID)**

Address: 4004\_7000h base + 1024h offset = 4004\_8024h

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

\* Notes:

- FAMID field: Device specific value.
- SUBFAMID field: Device specific value.
- SRAMSIZE field: Device specific value.
- REVID field: Device specific value.
- PINID field: Device specific value.

**SIM\_SDID field descriptions**

Field	Description
31–28 FAMID	<p>Family ID</p> <p>0011 KL33</p> <p>0001 KL13</p>

Table continues on the next page...

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

Field	Description
27–24 SUBFAMID	Kinetis Sub-Family ID Specifies the Kinetis sub-family of the device. 0011 KLx3 Subfamily
23–20 SERIESID	Kinetis Series ID Specifies the Kinetis family of the device. 0001 KL family
19–16 SRAMSIZE	System SRAM Size Specifies the size of the System SRAM 0011 4 KB 0100 8 KB
15–12 REVID	Device Revision Number Specifies the silicon implementation number for the device.
11–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PINID	Pincount Identification Specifies the pincount of the device. 0010 32-pin 0100 48-pin 0101 64-pin 0110 80-pin 1011 Custom pinout (WLCSP)

**12.3.7 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				SPI1	SPI0	0	VREF	CMPO	0	0	
W	[Shaded]										[Shaded]			[Shaded]	[Shaded]	
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	UART2	0	0	0	0	I2C1	I2C0	1		0				
W	[Shaded]			[Shaded]	[Shaded]	[Shaded]	[Shaded]					[Shaded]		[Shaded]		
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.
27–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SPI1	SPI1 Clock Gate Control Controls the clock gate to the SPI1 module. 0 Clock disabled 1 Clock enabled
22 SPI0	SPI0 Clock Gate Control Controls the clock gate to the SPI0 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.
20 VREF	VREF Clock Gate Control Controls the clock gate to the VREF module. 0 Clock disabled 1 Clock enabled
19 CMPO	Comparator Clock Gate Control 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 UART2	UART2 Clock Gate Control Controls the clock gate to the UART2 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.

*Table continues on the next page...*



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

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 I2C1	I2C1 Clock Gate Control  Controls the clock gate to the I <sup>2</sup> C1 module.  0 Clock disabled 1 Clock enabled
6 I2C0	I2C0 Clock Gate Control  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.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

**12.3.8 System Clock Gating Control Register 5 (SIM\_SCGC5)**

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0								LPUART1		LPUART0		0	0			
W	FLEXIO																
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	PORTE			PORTD	PORTC	PORTB	PORTA	1	0	0	0			1	LPTMR	
W																	
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0

**SIM\_SCGC5 field descriptions**

Field	Description
31 FLEXIO	FlexIO Module  This bit controls the clock gate to the FlexIO Module.  0 Clock disabled 1 Clock enabled

Table continues on the next page...

**SIM\_SCGC5 field descriptions (continued)**

Field	Description
30–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 LPUART1	LPUART1 Clock Gate Control  This bit controls the clock gate to the LPUART1 module.  0 Clock disabled 1 Clock enabled
20 LPUART0	LPUART0 Clock Gate Control  This bit controls the clock gate to the LPUART0 module.  0 Clock disabled 1 Clock enabled
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 PORTE	Port E Clock Gate Control  Controls the clock gate to the Port E module.  0 Clock disabled 1 Clock enabled
12 PORTD	Port D Clock Gate Control  Controls the clock gate to the Port D module.  0 Clock disabled 1 Clock enabled
11 PORTC	Port C Clock Gate Control  Controls the clock gate to the Port C module.  0 Clock disabled 1 Clock enabled
10 PORTB	Port B Clock Gate Control  Controls the clock gate to the Port B module.  0 Clock disabled 1 Clock enabled
9 PORTA	Port A Clock Gate Control  Controls the clock gate to the Port A module.  0 Clock disabled 1 Clock enabled
8–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

*Table continues on the next page...*

**SIM\_SCGC5 field descriptions (continued)**

Field	Description
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.
4–2 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 1.
0 LPTMR	Low Power Timer Access Control  Controls software access to the Low Power Timer module.  0 Access disabled 1 Access enabled

**12.3.9 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	
W	DAC0		RTC		ADC0	TPM2	TPM1	TPM0	PIT				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														
W															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 RTC	RTC Access Control

*Table continues on the next page...*

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

Field	Description
	Controls software access and interrupts to the RTC module. 0 Access and interrupts disabled 1 Access and interrupts enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 ADC0	ADC0 Clock Gate Control Controls the clock gate to the ADC0 module. 0 Clock disabled 1 Clock enabled
26 TPM2	TPM2 Clock Gate Control Controls the clock gate to the TPM2 module. 0 Clock disabled 1 Clock enabled
25 TPM1	TPM1 Clock Gate Control Controls the clock gate to the TPM1 module. 0 Clock disabled 1 Clock enabled
24 TPM0	TPM0 Clock Gate Control Controls the clock gate to the TPM0 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
22–19 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.
14–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

*Table continues on the next page...*

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

Field	Description
1 DMAMUX	DMA Mux Clock Gate Control Controls the clock gate to the DMA Mux module. 0 Clock disabled 1 Clock enabled
0 FTF	Flash Memory Clock Gate Control Controls the clock gate to the flash memory. Flash reads are still supported while the flash memory is clock gated, but entry into low power modes is blocked. 0 Clock disabled 1 Clock enabled

**12.3.10 System Clock Gating Control Register 7 (SIM\_SCGC7)**

Address: 4004\_7000h base + 1040h offset = 4004\_8040h

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

**SIM\_SCGC7 field descriptions**

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 DMA	DMA Clock Gate Control Controls the clock gate to the DMA module. 0 Clock disabled 1 Clock enabled
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

### 12.3.11 System Clock Divider Register 1 (SIM\_CLKDIV1)

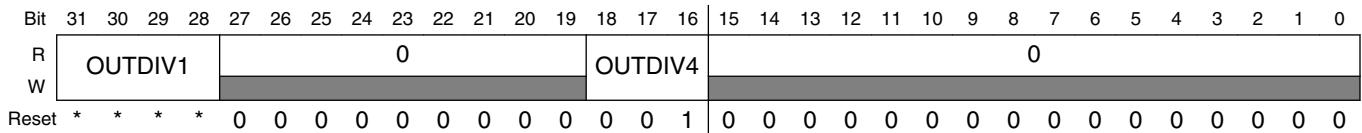
**NOTE**

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

**NOTE**

Reset value loaded during System Reset from FTFA\_FOPT[LPBOOT] (See [Table 6-2](#)).

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



\* Notes:

- OUTDIV1 field: The reset value depends on the FTFA\_FOPT[LPBOOT]. It is loaded with 0000 (divide by 1), 0001 (divide by 2), 0011 (divide by 4, or 0111 (divide by 8).

#### SIM\_CLKDIV1 field descriptions

Field	Description
31–28 OUTDIV1	<p>Clock 1 Output Divider value</p> <p>Sets the divide value for the core/system clock, as well as the bus/flash clocks. At the end of reset, it is loaded with 0000 (divide by one), 0001 (divide by two), 0011 (divide by four), or 0111 (divide by eight) depending on the setting of the FTFA_FOPT[LPBOOT] (See <a href="#">Table 6-2</a>).</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>
27–19 Reserved	<p>This field is reserved.                      This read-only field is reserved and always has the value 0.</p>

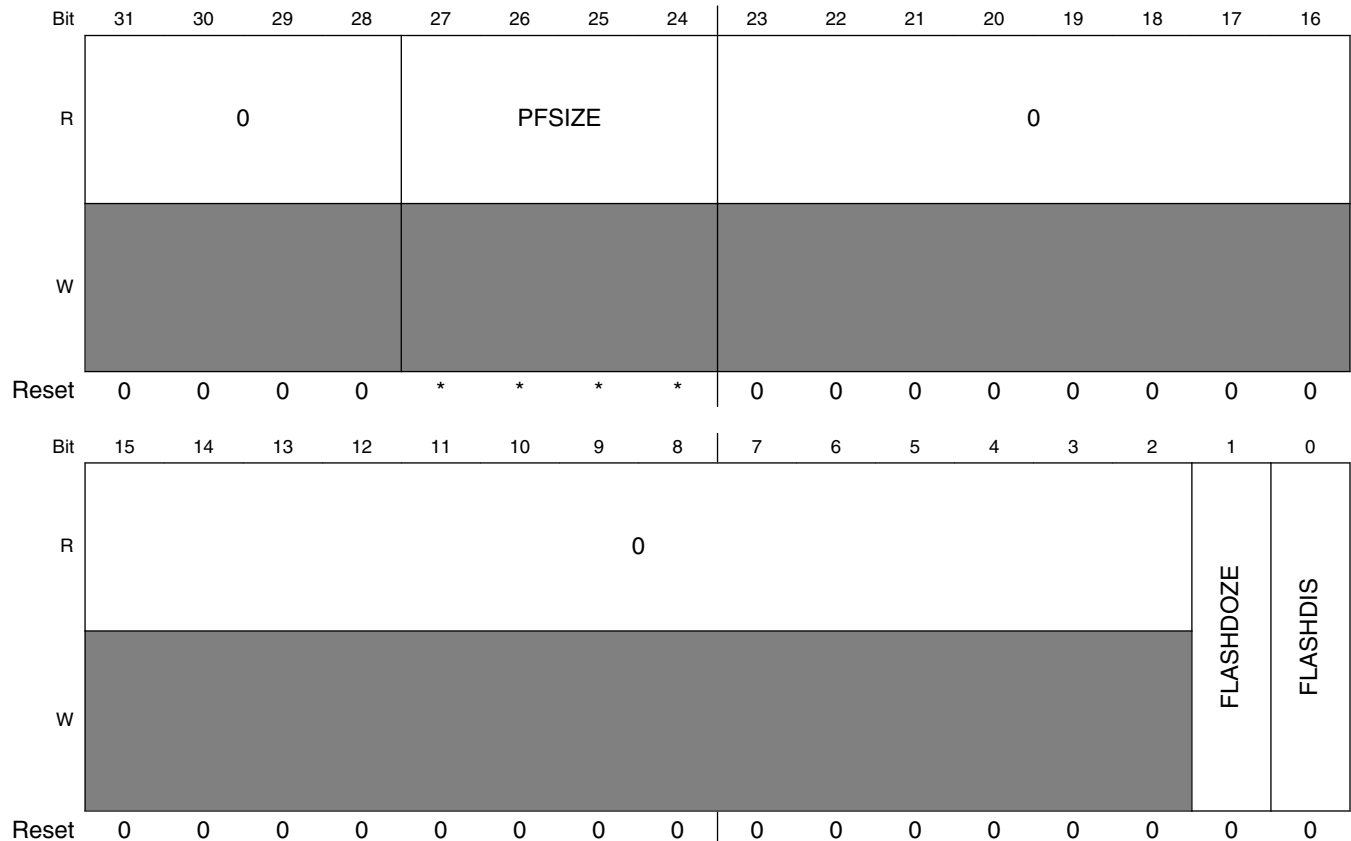
Table continues on the next page...

**SIM\_CLKDIV1 field descriptions (continued)**

Field	Description
18–16 OUTDIV4	<p>Clock 4 Output Divider value</p> <p>Sets the divide value for the bus and flash clock and is in addition to the System clock divide ratio. At the end of reset, it is loaded with 0001 (divide by 2).</p> <p>000 Divide-by-1.                      001 Divide-by-2.                      010 Divide-by-3.                      011 Divide-by-4.                      100 Divide-by-5.                      101 Divide-by-6.                      110 Divide-by-7.                      111 Divide-by-8.</p>
Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>

**12.3.12 Flash Configuration Register 1 (SIM\_FCFG1)**

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



- \* Notes:
- PFSIZE field: Device specific value.

## 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  Specifies the amount of program flash memory available on the device. Undefined values are reserved.  0000 8 KB of program flash memory, 1 KB protection region 0001 16 KB of program flash memory, 1 KB protection region 0011 32 KB of program flash memory, 1 KB protection region 0101 64 KB of program flash memory, 2 KB protection region 1111 64 KB of program flash memory, 2 KB protection region
23–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FLASHDOZE	Flash Doze  When set, flash memory is disabled for the duration of Doze mode. This field must be clear during VLP modes. The flash will be automatically enabled again at the end of Doze mode so interrupt vectors do not need to be relocated out of flash memory. The wake-up time from Doze mode is extended when this field is set. An attempt by the DMA or other bus master to access the flash memory when the flash is disabled will result in a bus error.  0 Flash remains enabled during Doze mode. 1 Flash is disabled for the duration of Doze mode.
0 FLASHDIS	Flash Disable  Flash accesses are disabled (and generate a bus error) and the flash memory is placed in a low-power state. This field should not be changed during VLP modes. Relocate the interrupt vectors out of Flash memory before disabling the Flash.  0 Flash is enabled. 1 Flash is disabled.



### 12.3.13 Flash Configuration Register 2 (SIM\_FCFG2)

This is read only register, any write to this register will cause transfer error.

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	0						
W	[Greyed out]															
Reset	0	*	*	*	*	*	*	*	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:

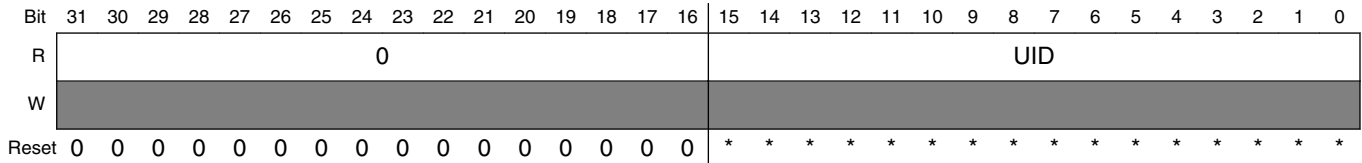
- MAXADDR0 field: Device specific value indicating amount of implemented flash.

#### 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 lock This field concatenated with 13 trailing zeros indicates the first invalid address of program flash. For example, if MAXADDR0 = 0x10, the first invalid address of program flash is 0x0002_0000. This would be the MAXADDR0 value for a device with 128 KB program flash.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
22–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

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

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



\* Notes:

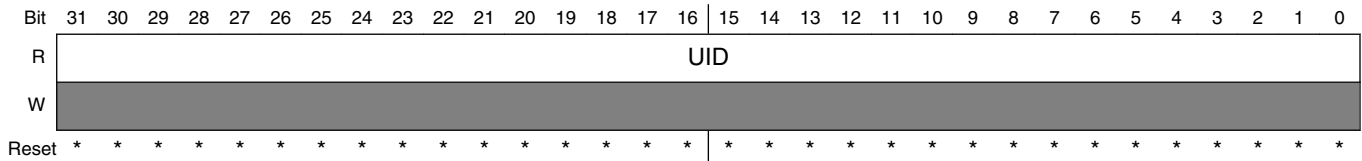
- UID field: Device specific value.

#### SIM\_UIDMH field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
UID	Unique Identification Unique identification for the device.

### 12.3.15 Unique Identification Register Mid Low (SIM\_UIDML)

Address: 4004\_7000h base + 105Ch offset = 4004\_805Ch



\* Notes:

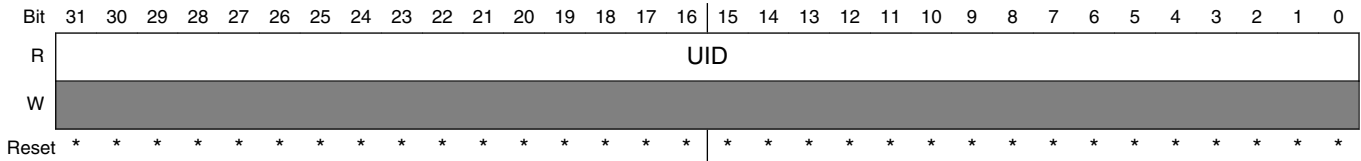
- UID field: Device specific value.

#### SIM\_UIDML field descriptions

Field	Description
UID	Unique Identification Unique identification for the device.

### 12.3.16 Unique Identification Register Low (SIM\_UIDL)

Address: 4004\_7000h base + 1060h offset = 4004\_8060h



\* Notes:

- UID field: Device specific value.

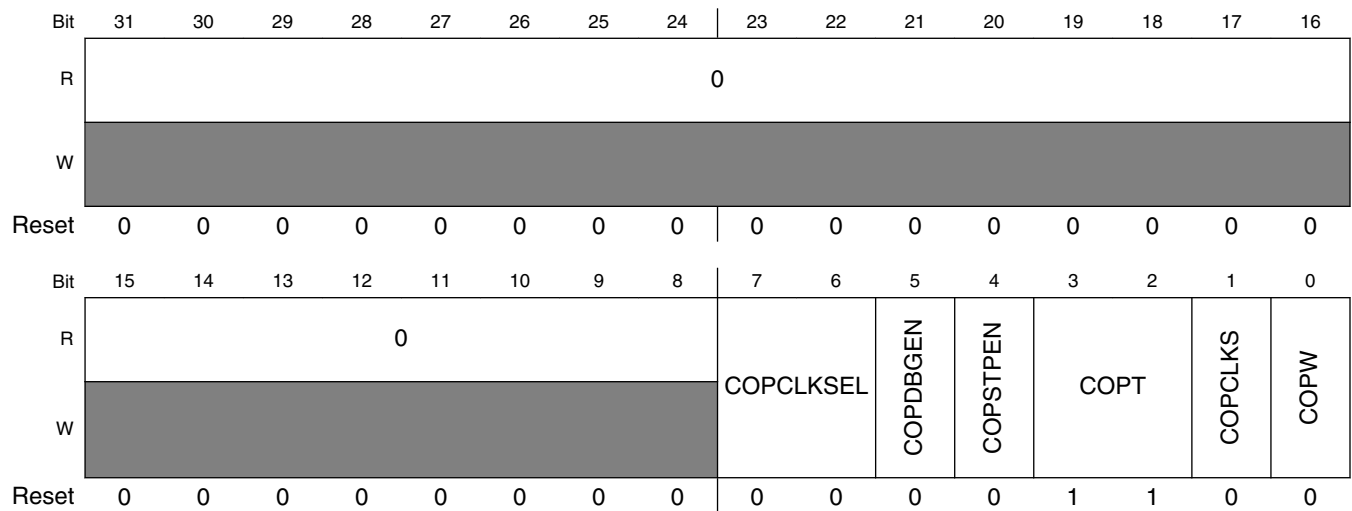
#### SIM\_UIDL field descriptions

Field	Description
UID	Unique Identification Unique identification for the device.

### 12.3.17 COP Control Register (SIM\_COPC)

All of the bits in this register can be written only once after a reset, writing this register will also reset the COP counter.

Address: 4004\_7000h base + 1100h offset = 4004\_8100h



#### SIM\_COPC 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...

## SIM\_COPC field descriptions (continued)

Field	Description
7–6 COPCLKSEL	<p>COP Clock Select</p> <p>This write-once field selects the clock source of the COP watchdog.</p> <p>00 LPO clock (1 kHz) 01 MCGIRCLK 10 OSCERCLK 11 Bus clock</p>
5 COPDBGEN	<p>COP Debug Enable</p> <p>0 COP is disabled and the counter is reset in Debug mode 1 COP is enabled in Debug mode</p>
4 COPSTPEN	<p>COP Stop Enable</p> <p>0 COP is disabled and the counter is reset in Stop modes 1 COP is enabled in Stop modes</p>
3–2 COPT	<p>COP Watchdog Timeout</p> <p>This write-once field selects the timeout period of the COP. COPT along with the COPCLKS field define the COP timeout period.</p> <p>00 COP disabled 01 COP timeout after <math>2^5</math> cycles for short timeout or <math>2^{13}</math> cycles for long timeout 10 COP timeout after <math>2^8</math> cycles for short timeout or <math>2^{16}</math> cycles for long timeout 11 COP timeout after <math>2^{10}</math> cycles for short timeout or <math>2^{18}</math> cycles for long timeout</p>
1 COPCLKS	<p>COP Clock Select</p> <p>This write-once field selects between a short timeout or a long timeout, the COP clock source is configured by COPCLKSEL.</p> <p>0 COP configured for short timeout 1 COP configured for long timeout</p>
0 COPW	<p>COP Windowed Mode</p> <p>Windowed mode is supported for all COP clock sources, but only when the COP is configured for a long timeout. The COP window is opened three quarters through the timeout period and will generate a system reset if the COP is serviced outside of that time.</p> <p>0 Normal mode 1 Windowed mode</p>

### 12.3.18 Service COP (SIM\_SRVCOP)

This is write only register, any read to this register will cause transfer error.

Address: 4004\_7000h base + 1104h offset = 4004\_8104h

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																SRVCOP															
W	Reserved																SRVCOP															
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\_SRVCOP field descriptions

Field	Description
31–8 Reserved	This field is reserved.
SRVCOP	Service COP Register  Write 0x55 and then 0xAA (in that order) to reset the COP timeout counter, writing any other value will generate a system reset.

## 12.4 Functional description

See [Introduction](#) section.

### 12.4.1 COP watchdog operation

The COP watchdog is intended to force a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), the application software must reset the COP counter periodically. If the application program gets lost and fails to reset the COP counter before it times out, a system reset is generated to force the system back to a known starting point.

After any reset, the COP watchdog is enabled. If the COP watchdog is not used in an application, it can be disabled by clearing SIM\_COPC[COPT].

The COP counter is reset by writing 0x55 and 0xAA (in that order) to the address of the SIM's Service COP (SRVCOP) register during the selected timeout period. Writes do not affect the data in the SRVCOP register. As soon as the write sequence is complete, the

COP timeout period is restarted. If the program fails to perform this restart during the timeout period, the microcontroller resets. Also, if any value other than 0x55 or 0xAA is written to the SRVCOP register, the microcontroller immediately resets.

SIM\_COPC[COPCLKS] and SIM\_COPCTRL[COPCLKSEL] select the timeout duration and clock source used for the COP timer. The clock source options are either the bus clock, 8 MHz/2 MHz IRC, external crystal or an internal 1 kHz clock source. With each clock source, the associated timeouts are controlled by SIM\_COPC[COPT] and SIM\_COPC[COPCLKS]. The following table summarizes the control functions of SIM\_COPCTRL[COPCLKS] and SIM\_COPC[COPCLKSEL] and SIM\_COPC[COPT] fields. The COP watchdog defaults to operation from the 1 kHz clock source and the longest timeout is 2<sup>10</sup> cycles.

**Table 12-1. COP configuration options**

Control bits			Clock source	COP window opens (SIM_COPC[COPW]=1)	COP overflow count
SIM_COPC[COPCLKSEL]	SIM_COPC[COPCLKS]	SIM_COPC[COPT]			
N/A	N/A	00	N/A	N/A	COP is disabled.
00	0	01	1 kHz	N/A	2 <sup>5</sup> cycles (32ms)
	1			6,144 cycles	2 <sup>13</sup> cycles (8192ms)
00	0	10	1 kHz	N/A	2 <sup>8</sup> cycles (256ms)
	1			49,152 cycles	2 <sup>16</sup> cycles (65536ms)
00	0	11	1 kHz	N/A	2 <sup>10</sup> cycles (1024 ms)
	1			196,608 cycles	2 <sup>18</sup> cycles (262144ms)
01	0	01	8/2 MHz	N/A	2 <sup>5</sup> cycles
	1			6,144 cycles	2 <sup>13</sup> cycles
01	0	10	8/2 MHz	N/A	2 <sup>8</sup> cycles
	1			49,152 cycles	2 <sup>16</sup> cycles
01	0	11	8/2 MHz	N/A	2 <sup>10</sup> cycles
	1			196,608 cycles	2 <sup>18</sup> cycles
10	0	01	crystal	N/A	2 <sup>5</sup> cycles
	1			6,144 cycles	2 <sup>13</sup> cycles
10	0	10	crystal	N/A	2 <sup>8</sup> cycles
	1			49,152 cycles	2 <sup>16</sup> cycles
10	0	11	crystal	N/A	2 <sup>10</sup> cycles
	1			196,608 cycles	2 <sup>18</sup> cycles
01	0	01	bus	N/A	2 <sup>5</sup> cycles
	1			6,144 cycles	2 <sup>13</sup> cycles
01	0	10	bus	N/A	2 <sup>8</sup> cycles

Table continues on the next page...

**Table 12-1. COP configuration options (continued)**

Control bits			Clock source	COP window opens (SIM_COPC[COPW]=1)	COP overflow count
SIM_COPC[COPCLK SEL]	SIM_COPC[COPCLK S]	SIM_COPC[COP T]			
	1			49,152 cycles	2 <sup>16</sup> cycles
01	0	11	bus	N/A	2 <sup>10</sup> cycles
	1			196,608 cycles	2 <sup>18</sup> cycles

After the long timeout (COPCLKS = 1) is selected, windowed COP operation is available by setting SIM\_COPC[COPW]. In this mode, writes to SIM\_SRVCOP to clear the COP timer must occur in the last 25% of the selected timeout period. A premature write immediately resets the chip. When the short timeout (COPCLKS =0) is selected, windowed COP operation is not available.

The COP counter is initialized by the first writes to SIM\_COPC and after any system reset. Subsequent writes to SIM\_COPC have no effect on COP operation. Even if an application uses the reset default settings of SIM\_COPC[COPT], SIM\_COPC[COPCLKS], SIM\_COPC[COPCLKSEL], and SIM\_COPC[COPW] fields, the user should write to the write-once SIM\_COPC register during reset initialization to lock in the settings. This approach prevents accidental changes if the application program becomes lost.

The write to SIM\_SRVCOP that services (clears) the COP counter should not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

If the selected clock is not the 1 kHz clock source, the COP counter does not increment while the microcontroller is in Debug mode or while the system is in Stop (including VLPS or LLS) mode. The COP counter resumes when the microcontroller exits Debug or Stop mode.

The COP counter is re-initialized to 0 upon entry to either Debug mode or Stop (including VLPS or LLS) mode. The counter begins from 0 upon exit from Debug mode or Stop mode.

The COP counter can also be configured to continue incrementing during Debug mode or Stop (including VLPS) mode if either COPDBGEN or COPSTPEN are set respectively. When the selected clock is the bus clock and COPSTEN bit is set, the COP counter cannot increment during Stop modes, however the COP counter is not reset to 0.

Regardless of the clock selected, the COP is disabled when the chip enters a VLLSx mode. Upon a reset that wakes the chip from the VLLSx mode, the COP is reinitialized and enabled as for any reset.





# Chapter 13

## Kinetis ROM Bootloader

### 13.1 Chip-Specific Information

This device has various peripherals (UART, I2C, SPI ) supported by the Kinetis ROM Bootloader. To use an interface for bootloader communications, the peripheral must be enabled in the BCA, as shown in [Table 13-3](#). If the BCA is invalid (such as all 0xFF bytes), then all peripherals will be enabled by default. The next table shows the pads used by the Kinetis ROM Bootloader.

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

Port	Signal
PTA2	LPUART0_TX
PTA1	LPUART0_RX
PTB0	I2C0_SCL
PTB1	I2C0_SDA
PTC4	SPI0_SS_b
PTC7	SPI0_MISO
PTC6	SPI0_MOSI
PTC5	SPI0_SCK

### 13.2 Introduction

The Kinetis bootloader is the program residing in the on-chip read-only memory (ROM) of a Kinetis microcontroller device. There is hardware logic in place at boot time that either starts execution of an embedded image available on the internal flash memory, or starts the execution of the Kinetis Bootloader from on-chip ROM.

The Kinetis Bootloader's main task is to provision the internal flash memory with an embedded firmware image during manufacturing, or at any time during the life of the Kinetis device. The Kinetis Bootloader does the provisioning by acting as a slave device, and listening to various peripheral ports where a master can start communication.

For the Kinetis device, the Kinetis Bootloader can interface with I2C, SPI, and LPUART 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 Bootloader. Regardless of the host/master (PC or embedded host), the Kinetis Bootloader always uses a command protocol to communicate with that host/master. Commands are provided to write to memory (internal flash or RAM), erase flash, and get/set bootloader options and property values. The host application can query the set of available commands.

On start-up, the bootloader reads optional configuration parameters from a fixed area on flash called the bootloader configuration area (BCA). These parameters can be modified by the write memory command or by downloaded flash image. BCA parameters include configuration data such as enabled peripherals, peripheral-specific settings, etc.

This chapter describes Kinetis Bootloader features, functionality, command structure and which peripherals are supported.

Features supported by the Kinetis Bootloader in Kinetis ROM:

- Supports I2C, SPI, and LPUART peripheral interfaces
- Automatic detection of the active peripheral
- Ability to disable any peripheral
- LPUART peripheral implements autobaud
- Common packet-based protocol for all peripherals
- Packet error detection and retransmission
- Flash-resident configuration options
- Fully supports internal flash security, including ability to mass erase or unlock security via the backdoor key
- Protection of RAM used by the bootloader while it is running
- Provides command to read properties of the device, such as flash and RAM size
- Multiple options for executing the bootloader either at system start-up or under application control at runtime

**Table 13-2. Commands supported by the Kinetis Bootloader in ROM**

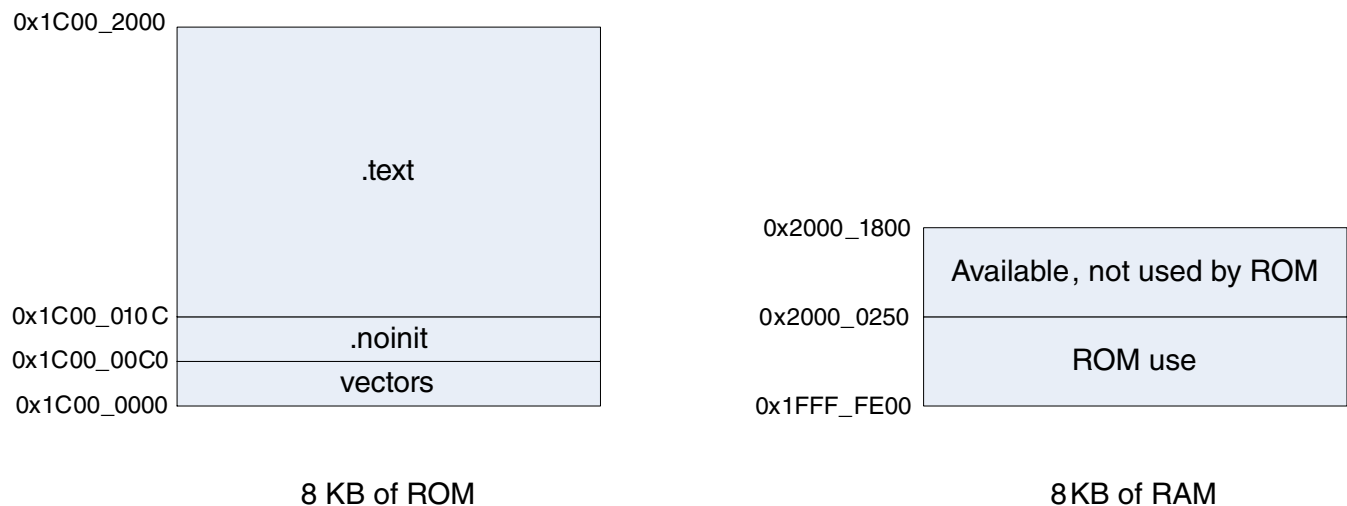
Command	Description	When flash security is enabled, then this command is
Execute	Run user application code that never returns control to the bootloader	Not supported
FlashEraseAll	Erase the entire flash array	Not supported
FlashEraseRegion	Erase a range of sectors in flash	Not supported
WriteMemory	Write data to memory	Not supported
FlashSecurityDisable	Attempt to unlock flash security using the backdoor key	Supported
GetProperty	Get the current value of a property	Supported
Reset	Reset the chip	Supported
SetProperty	Attempt to modify a writable property	Supported
FlashEraseAllUnsecure	Erase the entire flash array, including protected sectors	Supported

## 13.3 Functional Description

The following sub-sections describe the Kinetis Bootloader in KLx3 ROM functionality.

### 13.3.1 Memory Maps

While executing, the Kinetis Bootloader uses ROM and RAM memory.

**Figure 13-1. Kinetis Bootloader ROM/RAM Memory Maps**

### 13.3.2 The Kinetis Bootloader Configuration Area (BCA)

The Kinetis Bootloader reads data from the Bootloader Configuration Area (BCA) to configure various features of the bootloader. The BCA resides in flash memory at offset 0x3C0, and provides all of the parameters needed to configure the Kinetis Bootloader operation. For uninitialized flash, the Kinetis Bootloader uses a predefined default configuration. A host application can use the Kinetis Bootloader to program the BCA for use during subsequent initializations of the bootloader.

**Table 13-3. Configuration Fields for the Kinetis Bootloader**

Offset	Size (bytes)	Configuration Field	Description
0x00 - 0x03	4	tag	Magic number to verify bootloader configuration is valid. Must be set to 'kcfg'.
0x04 - 0x07	4	-	Reserved in KLx3
0x08 - 0x0B	4	-	Reserved in KLx3
0x0C - 0x0F	4	-	Reserved in KLx3
0x10	1	enabledPeripherals	Bitfield of peripherals to enable. bit 0 LPUART bit 1 I2C bit 2 SPI  Kinetis bootloader will enable the peripheral if corresponding bit is set to 1.
0x11	1	i2cSlaveAddress	If not 0xFF, used as the 7-bit I2C slave address. If 0xFF, defaults to 0x10 for I2C slave address
0x12 - 0x13	2	peripheralDetectionTimeout	Timeout in milliseconds for active peripheral detection. If 0xFFFF, defaults to 5 seconds.
0x14 - 0x15	2	-	Reserved
0x16 - 0x17	2	-	Reserved in
0x18 - 0x1B	4	-	Reserved in
0x1C	1	clockFlags	See <a href="#">Table 13-5</a> , clockFlags Configuration Field
0x1D	1	clockDivider	Inverted value of the divider to use for core and bus clocks when in high speed mode
0x1F	1	pad byte	N/A
0x24	4	Reserved	-

The first configuration field 'tag' is a tag value or magic number. The tag value must be set to 'kcfg' for the bootloader configuration data to be recognized as valid. If tag-field verification fails, then the Kinetis Bootloader assumes that the flash is not initialized and uses a predefined default configuration. The tag value is treated as a character string, so bytes 0-3 must be set as shown in the table.

**Table 13-4. tag Configuration Field**

Offset	tag Byte Value
0	'k' (0x6B)
1	'c' (0x63)
2	'f' (0x66)
3	'g' (0x67)

The flags in the clockFlags configuration field are enabled if the corresponding bit is cleared (0).

**Table 13-5. clockFlags Configuration Field**

Bit	Flag	Description
0	HighSpeed	Enable high speed mode (i.e., 48 MHz).
1 - 7	Reserved	Not used in KLx3 ROM

### 13.3.3 Start-up Process

Any of the following conditions will force the hardware to start the Kinetis Bootloader:

- BOOTSRC\_SEL field of FOPT register is set to either 0b11 or 0b10. This forces the ROM to run out of reset.
- The BOOTCFG0 pin is asserted. The pin must be configured as BOOTCFG0 by setting the BOOTPIN\_OPT bit of FOPT to 0.
- A user applications running on flash or RAM calls into the Kinetis Bootloader entry point address in ROM, to start Kinetis Bootloader execution.

The BOOTSRC\_SEL bits (FOPT register, FOPT [7:6]) determine the boot source. The FOPT register is located in the flash configuration field at address 0x40D in the flash memory array. For a complete list of options, see the Boot options section in the Reset and Boot chapter. If BOOTSRC\_SEL is set to 0b11 or 0b10, then the device will boot to ROM out of reset. Flash memory defaults to all 1s when erased, so a blank chip will automatically boot to ROM.

The BOOTCFG0 pin is shared with the NMI pin, with NMI being the default usage. Regardless of whether the NMI pin is enabled or not, the NMI functionality is disabled if the ROM is executed out of reset, for as long as the ROM is running.

When the ROM is executed out of reset, vector fetches from the CPU are redirected to the ROM's vector table in ROM memory at offset 0x1C00\_0000. This ensures that any exceptions will be handled by the ROM.

After the Kinetis Bootloader has started, the following procedure starts bootloader operations:

1. The RCM\_MR [FORCEROM] bits are set, so that the device will reboot back into the ROM if/when the device is reset.
2. Initializes the bootloader's .data and .bss sections.
3. Reads bootloader configuration data from flash at address 0x3C0. The configuration data is only used if the tag field is set to the expected 'kcfg' value. If the tag is incorrect, then the configuration values are set to default, as if the data was all 0xFF bytes.
4. Clocks are configured. See the [Clock Configuration](#) section.
5. Enabled peripherals are initialized.
6. The bootloader waits for communication to begin on a peripheral.
  - If detection times out, then the bootloader jumps to the user application in flash. See [Bootloader Exit state](#) section.
  - If communication is detected, then all inactive peripherals are shut down, and the command phase is entered.

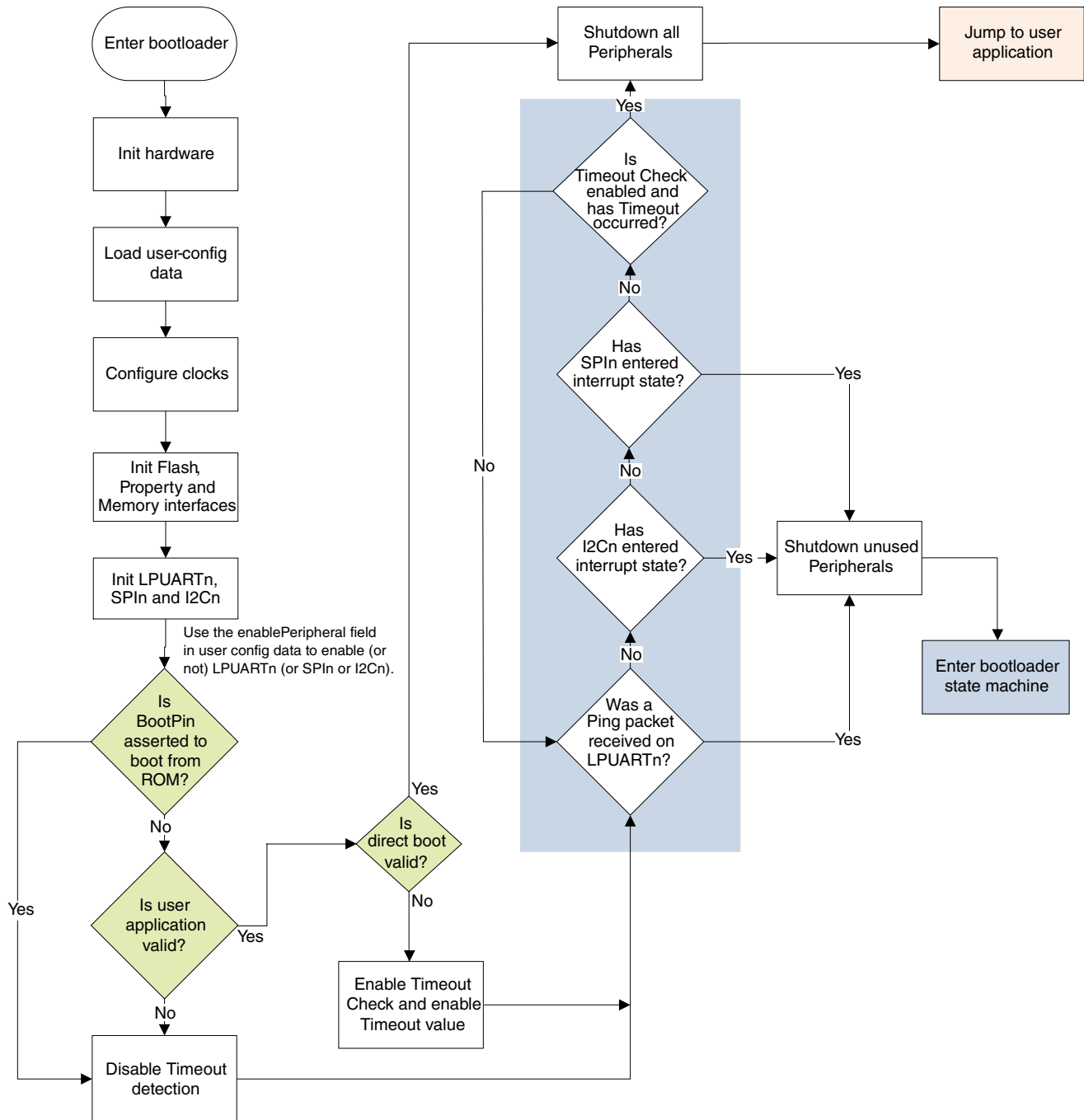


Figure 13-2. Kinetis Bootloader Start-up Flowchart

### 13.3.4 Clock Configuration

By default, the bootloader does not modify clocks. The Kinetis Bootloader in ROM will use the clock configuration of the chip out of reset unless the clock configuration bits in the FOPT register are cleared.

- Alternate clock configurations are supported, by setting fields in the Bootloader Configuration Area (BCA) shown in [Table 13-3](#).
- If the HighSpeed flag of the clockFlags configuration value is cleared, the bootloader will enable the internal 48 MHz reference clock.
- In high speed mode, the core and bus clock frequencies are determined by the clockDivider configuration value.
- The core clock divider is set directly from the inverted value of the clockDivider.
- The bus clock divider is set to 1, unless the resulting bus clock frequency would be greater than the maximum supported value. In this case, the bus clock divider is increased until the bus clock frequency is at or below the maximum.
- Note that the maximum baud rate of serial peripherals is related to the core and bus clock frequencies. To achieve the desired baud rates, high speed mode should be enabled in BCA.

### 13.3.5 Bootloader Entry Point

The Kinetis Bootloader provides a function (runBootloader) that a user application can call, to run the bootloader.

To get the address of the entry point, the user application reads the word containing the pointer to the bootloader API tree at offset 0x1C of the bootloader's vector table. The vector table is placed at the base of the bootloader's address range, which for the ROM is 0x1C00\_0000. Thus, the API tree pointer is at address 0x1C00\_001C.

The bootloader API tree is a structure that contains pointers to other structures, which have the function and data addresses for the bootloader. The bootloader entry point is always the first word of the API tree.

The prototype of the entry point is:

```
void run_bootloader(void * arg);
```

The arg parameter is currently unused, and is intended for future expansion (for example, passing options to the bootloader). To ensure future compatibility, a value of NULL should be passed for arg.

Example code to get the entry pointer address from the ROM and start the bootloader.

#### **NOTE**

This entry must be called in supervisor (privileged) mode.

```
// Variables
uint32_t runBootloaderAddress;

void (*runBootloader)(void * arg);
```



```
// Read the function address from the ROM API tree.
runBootloaderAddress = **(uint32_t **) (0x1c00001c);
runBootloader = (void (*)(void * arg))runBootloaderAddress;

// Start the bootloader.
runBootloader(NULL);
```

### 13.3.6 Bootloader Protocol

This section explains the general protocol for the packet transfers between the host and the Kinetis Bootloader. 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 bootloader ), then the data phase is part of the **original command**.
- If the data phase is **outgoing** (from bootloader to host), then the data phase is part of the **response command**.

#### NOTE

In all protocols (described in the next subsections), the Ack sent in response to a Command or Data packet can arrive at any time *before, during, or after* the Command/Data packet has processed.

#### 13.3.6.1 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)

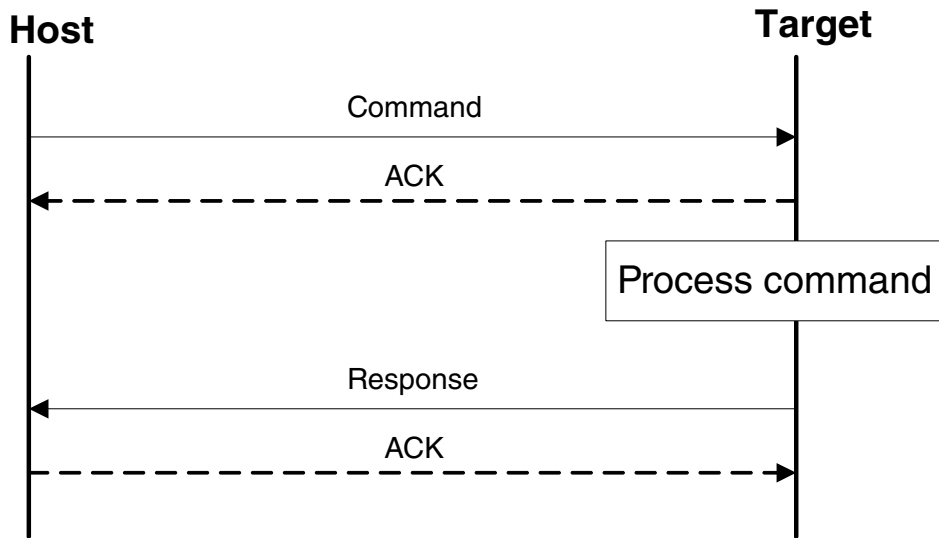
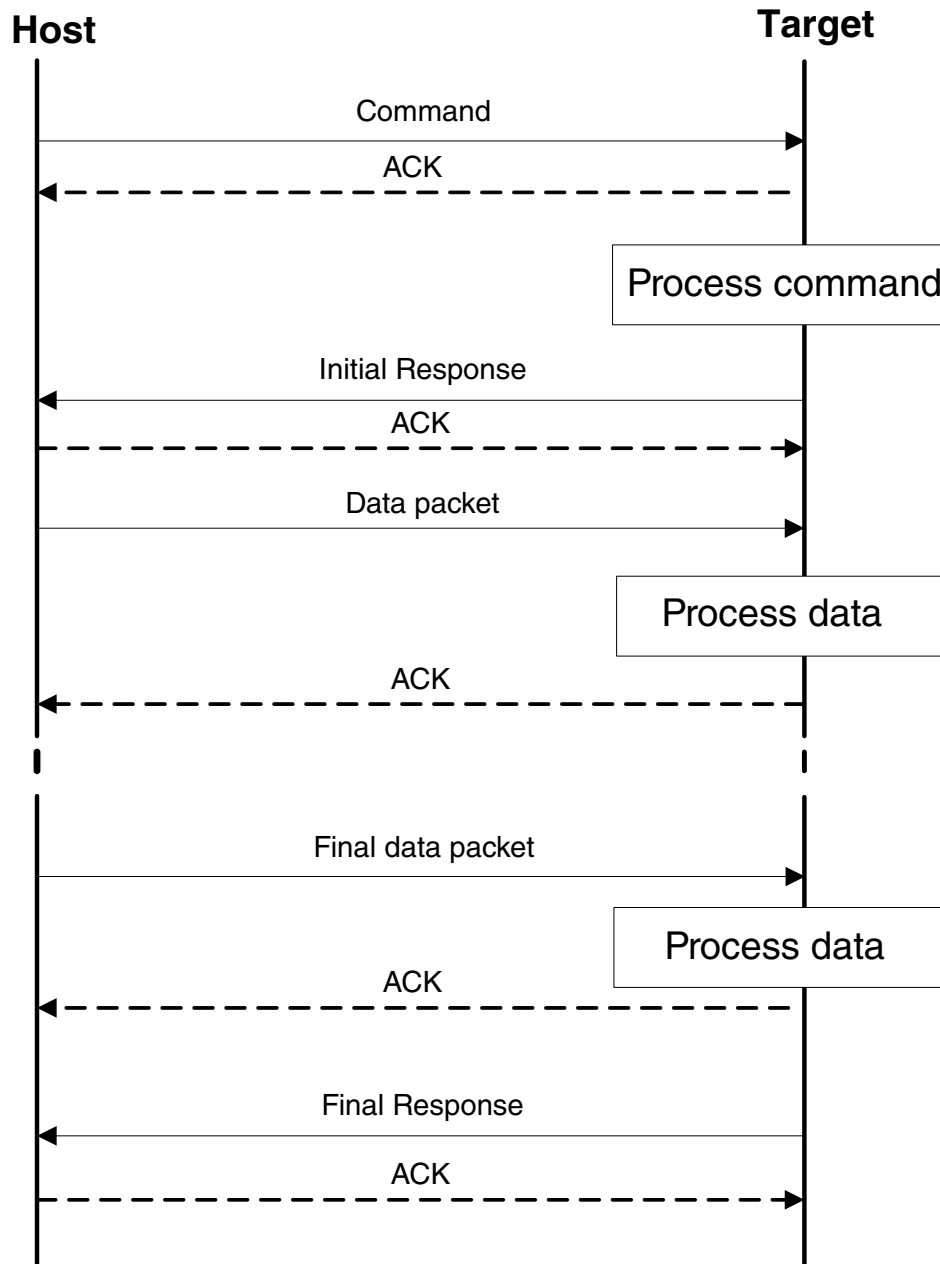


Figure 13-3. Command with No Data Phase

### 13.3.6.2 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.

### 13.3.6.3 Command with outgoing data phase

For KLx3, there is no command available with an outgoing data phase.

## 13.3.7 Bootloader Packet Types

The Kinetis Bootloader device works in slave mode. All data communication is initiated by a host, which is either a PC or an embedded host . The Kinetis Bootloader 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 Bootloader device."

There are 6 types of packets used in the device:

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

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

### 13.3.7.1 Ping packet

The Ping packet is the first packet sent from a host to the target (Kinetis Bootloader), to establish a connection on a selected peripheral. For a LPUART 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 13-6. Ping Packet Format**

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

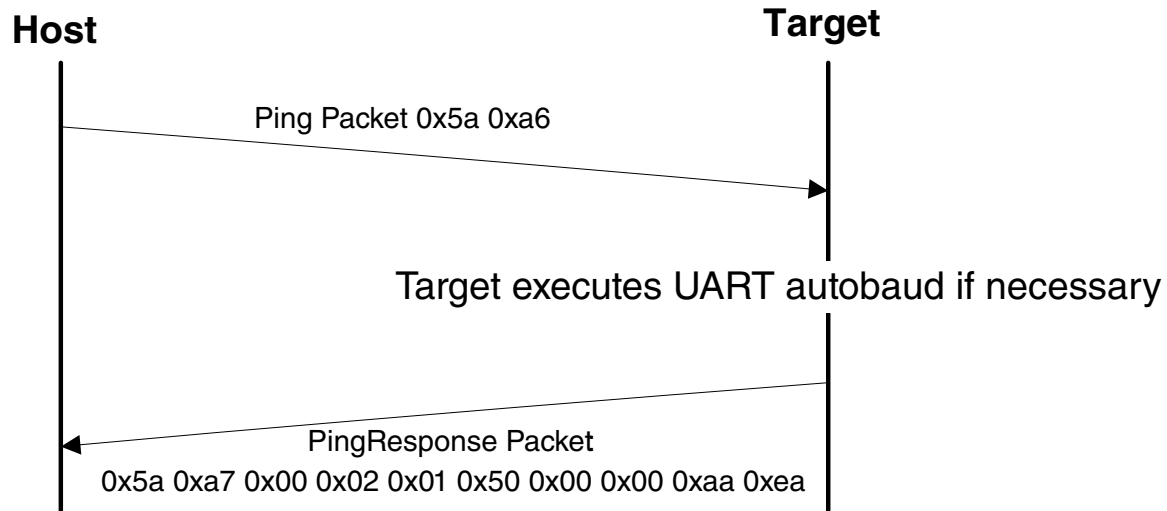


Figure 13-5. Ping Packet Protocol Sequence

### 13.3.7.2 Ping Response Packet

The target (Kinetis Bootloader) sends a Ping Response packet back to the host after receiving a Ping packet. If communication is over a LPUART 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 Bootloader).

Table 13-7. Ping Response Packet Format

Byte #	Value	Parameter
0	0x5A	start byte
1	0xA7	Ping response code
2		Protocol bugfix
3		Protocol minor
4		Protocol major
5		Protocol name = 'P' (0x50)
6		Options low
7		Options high
8		CRC16 low
9		CRC16 high

### 13.3.7.3 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 13-8. Framing Packet Format**

Byte #	Value	Parameter	
0	0x5A	start byte	
1		packetType	
2		length_low	Length is a 16-bit field that specifies the entire command or data packet size in bytes.
3		length_high	
4		crc16_low	This is a 16-bit field. The CRC16 value covers entire framing packet, including the start byte and command or data packets, but does not include the CRC bytes. See the CRC16 algorithm after this table.
5		crc16_high	
6 . . . n		Command or Data packet payload	

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

**Table 13-9. 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 13-10. 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.

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;
        }
    }
    return crc;
}

```

### 13.3.7.4 Command packet

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

**Table 13-11. 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 13-12. 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 13-13. Commands that are supported**

Command	Name
0x01	FlashEraseAll
0x02	FlashEraseRegion
0x03	Reserved
0x04	WriteMemory
0x05	Reserved
0x06	FlashSecurityDisable
0x07	GetProperty
0x08	Reserved
0x09	Execute
0x10	Reserved
0x11	Reserved
0x0A	Reserved
0x0B	Reset
0x0C	SetProperty
0x0D	FlashEraseAllUnsecure
0x0E	Reserved
0x0F	Reserved

**Table 13-14. Responses that are supported**

Response	Name
0xA0	GenericResponse
0xA7	GetPropertyResponse (used for sending responses to GetProperty command only)



**Flags:** Each command packet contains a Flag byte. Only bit 0 of the flag byte is used. If bit 0 of the flag byte is set to 1, then data packets will follow in the command sequence. The number of bytes that will be transferred in the data phase is determined by a command-specific parameter in the parameters array.

**ParameterCount:** The number of parameters included in the command packet.

**Parameters:** The parameters are word-length (32 bits). With the default maximum packet size of 32 bytes, a command packet can contain up to 7 parameters.

### 13.3.7.5 Data packet

The data packet carries just the data, either host sending data to target, or target sending data to host. The data transfer direction is determined by the last command sent from the host. The data packet is also wrapped within a framing packet, to ensure the correct packet data is received.

The contents of a data packet are simply the data itself. There are no other fields, so that the most data per packet can be transferred. Framing packets are responsible for ensuring that the correct packet data is received.

### 13.3.7.6 Response packet

The responses are carried using the same command packet format wrapped with framing packet data. Types of responses include:

- GenericResponse
- GetPropertyResponse

**GenericResponse:** After the Kinetis Bootloader has processed a command, the bootloader will send a generic response with status and command tag information to the host. The generic response is the last packet in the command protocol sequence. The generic response packet contains the framing packet data and the command packet data (with generic response tag = 0xA0) and a list of parameters (defined in the next section). The parameter count field in the header is always set to 2, for status code and command tag parameters.

**Table 13-15. GenericResponse Parameters**

Byte #	Parameter	Description
0 - 3	Status code	The Status codes are errors encountered during the execution of a command by the target (Kinetis Bootloader). If a command succeeds, then a kStatus_Success code is returned. <a href="#">Table 13-39</a> , Kinetis Bootloader Status Error Codes, lists the status codes returned to the host by the Kinetis Bootloader for KLx3 ROM.
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 13-16. 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.

### 13.3.8 Bootloader Command API

All Kinetis Bootloader 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 Kinetis Bootloader in KLx3 ROM, see [Table 13-2](#), Commands supported.
- For a list of status codes returned by the Kinetis Bootloader in KLx3 ROM, see [Table 13-39](#), Kinetis Bootloader Status Error Codes.

### 13.3.8.1 ReceiveSBFile command

The ReceiveSBFile command (ReceiveSbFile) will start the transfer of an SB file to the target. The command only specifies the size in bytes of the SB file that will be sent in the data phase. The SB file will be processed as it is received by the bootloader.

**Table 13-17. Parameters for ReceiveSBFile Command**

Byte #	Command
0 - 3	Byte count

**Data Phase:** The Receive SB file 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 Receive SB File command are received by the target.

**Response:** The target (Bootloader) will return a GenericResponse packet with a status code set to the kStatus\_Success upon successful execution of the command, or set to an appropriate error code.

### 13.3.8.2 Execute command

The execute command results in the bootloader setting the program counter to the code at the provided jump address, R0 to the provided argument, and a Stack pointer to the provided stack pointer address. Prior to the jump, the system is returned to the reset state.

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

**Table 13-18. Parameters for Execute Command**

Byte #	Command
0 - 3	Jump address
4 - 7	Argument word
8 - 11	Stack pointer address

The Execute command has no data phase.

**Response:** Before executing the Execute command, the target (Kinetis Bootloader) will validate the parameters and return a GenericResponse packet with a status code either set to kStatus\_Success or an appropriate error status code.

### 13.3.8.3 Reset command

The Reset command will result in bootloader resetting the chip.

The Reset command requires no parameters.

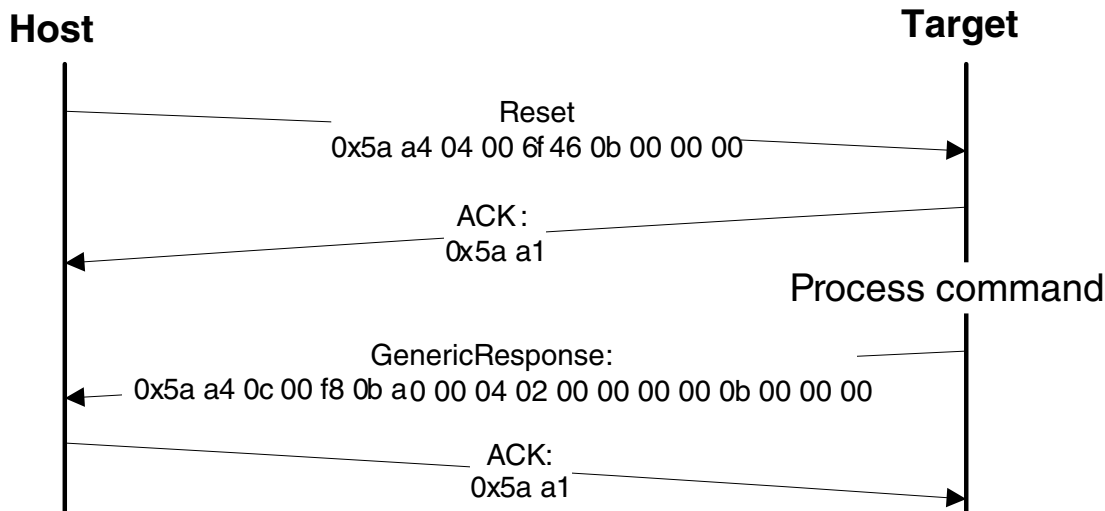


Figure 13-6. Protocol Sequence for Reset Command

Table 13-19. 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 Bootloader) will return a GenericResponse packet with status code set to kStatus\_Success, before resetting the chip.

### 13.3.8.4 GetProperty command

The GetProperty command is used to query the bootloader 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.

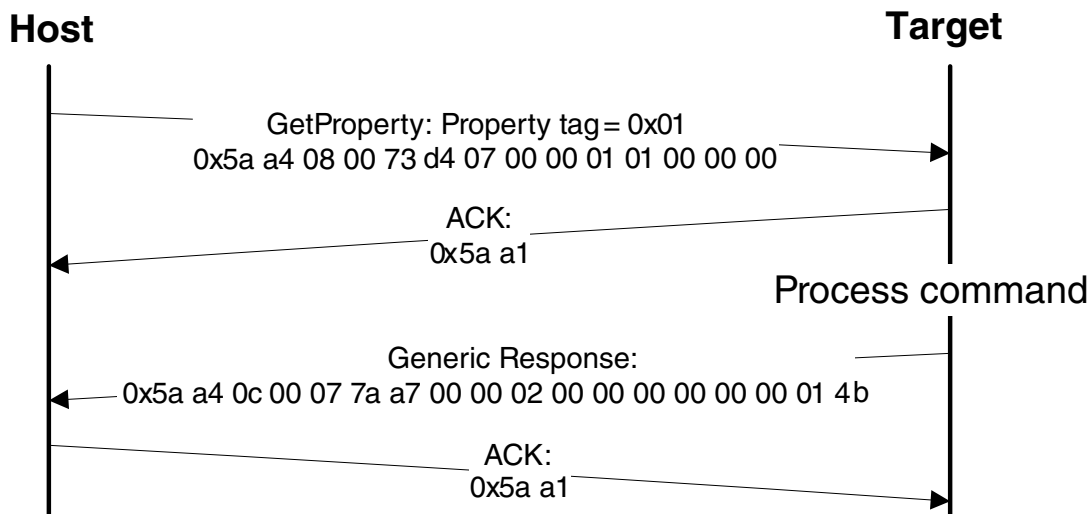
Properties are the defined units of data that can be accessed with the GetProperty or SetProperty commands. Properties may be read-only or read-write. All read-write properties are 32-bit integers, so they can easily be carried in a command parameter.

For a list of properties and their associated 32-bit property tags supported by the Kinetis Bootloader in KLx3 ROM, see [Table 13-35](#).

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

**Table 13-20. Parameters for GetProperty Command**

Byte #	Command
0 - 3	Property tag



**Figure 13-7. Protocol Sequence for GetProperty Command**

**Table 13-21. GetProperty Command Packet Format (Example)**

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

*Table continues on the next page...*

**Table 13-21. GetProperty Command Packet Format (Example) (continued)**

GetProperty	Parameter	Value
	crc16	0x73 0xD4
Command packet	commandTag	0x07 – GetProperty
	flags	0x00
	reserved	0x00
	parameterCount	0x01
	propertyTag	0x00000001 - CurrentVersion

The GetProperty command has no data phase.

**Response:** In response to a GetProperty command, the target will send a GetPropertyResponse packet with the response tag set to 0xA7. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property value(s). The next table shows an example of a GetPropertyResponse packet.

**Table 13-22. 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

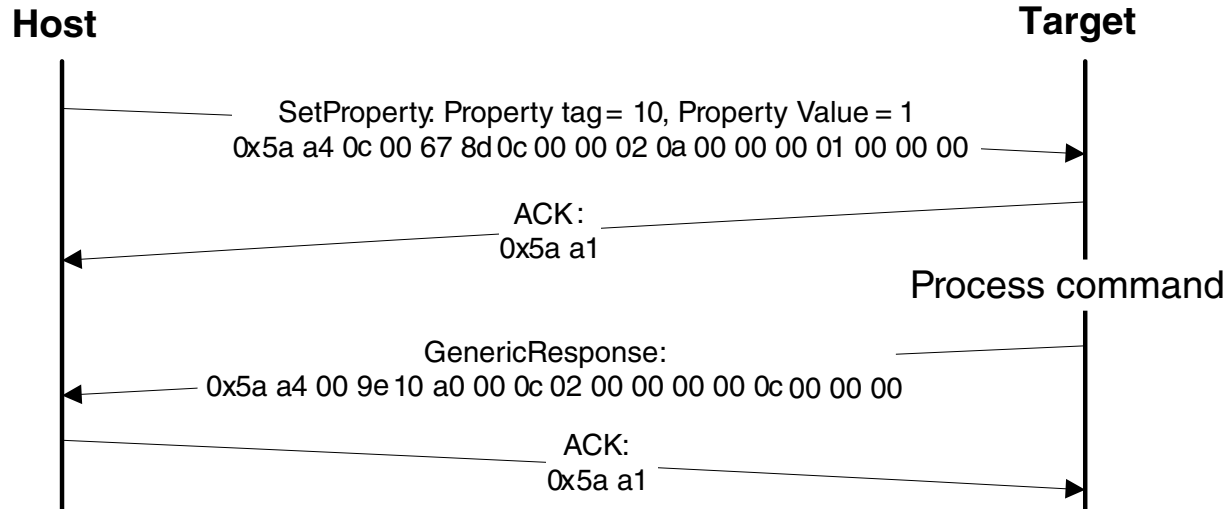
### 13.3.8.5 SetProperty command

The SetProperty command is used to change or alter the values of the properties or options in the Kinetis Bootloader ROM. 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 Bootloader will return an error.

The property tag and the new value to set are the 2 parameters required for the SetProperty command.

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

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



**Figure 13-8. Protocol Sequence for SetProperty Command**

**Table 13-24. 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 Bootloader) will return a GenericResponse packet with one of following status codes:

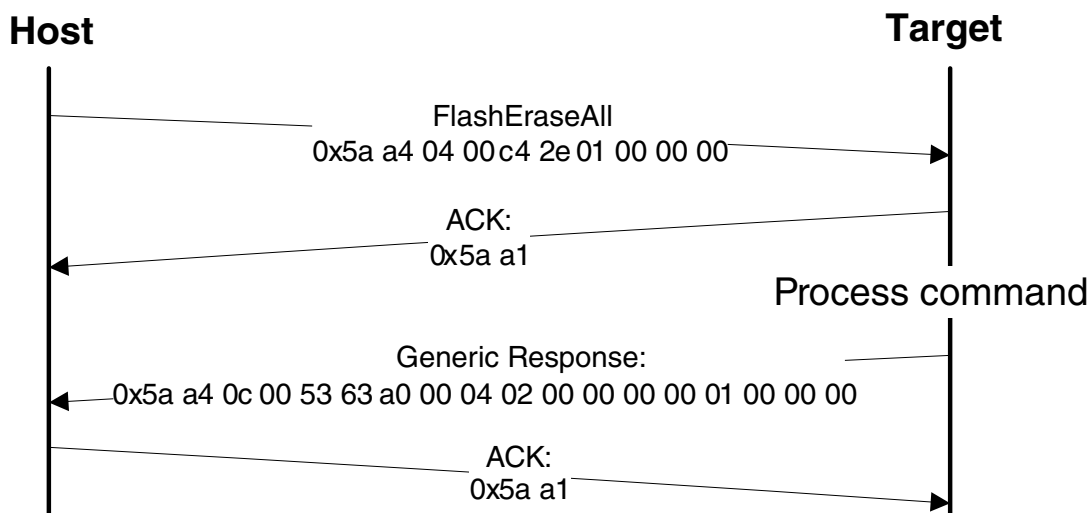
**Table 13-25. SetProperty Response Status Codes**

Status Code
kStatus_Success
kStatus_ReadOnly
kStatus_UnknownProperty
kStatus_InvalidArgument

### 13.3.8.6 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 13-26. 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 Bootloader ) 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.

### 13.3.8.7 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory or a specified range of flash within the connected SPI flash devices.

The start address and number of bytes are the 2 parameters required for the FlashEraseRegion command. The start and byte count parameters must be 4-byte aligned ([1:0] = 00), or the FlashEraseRegion command will fail and return kStatus\_FlashAlignmentError (0x101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command will fail and return kStatus\_FlashAddressError (0x102). If any part of the region specified is protected, the FlashEraseRegion command will fail and return kStatus\_MemoryRangeInvalid (0x10200).

**Table 13-27. Parameters for FlashEraseRegion Command**

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

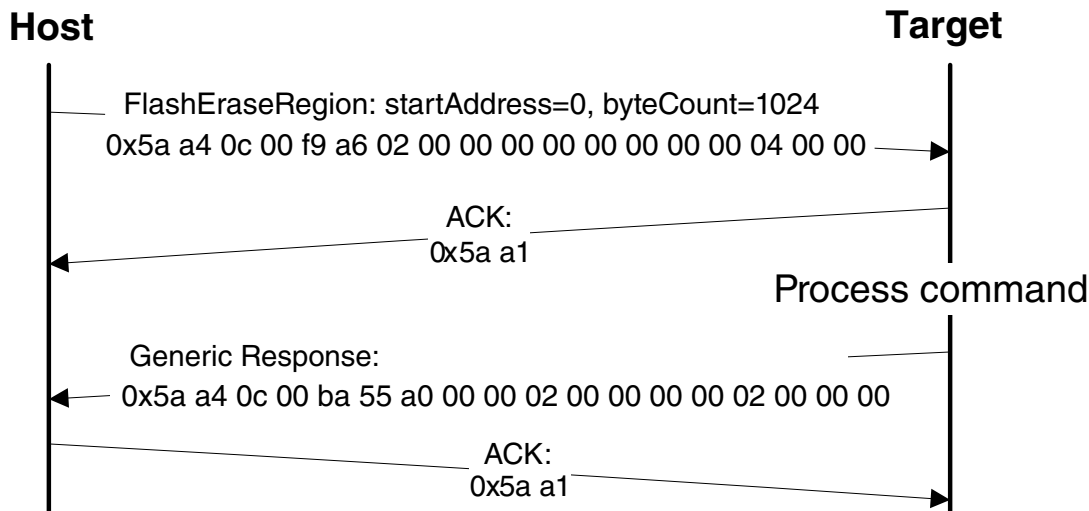


Figure 13-10. Protocol Sequence for FlashEraseRegion Command

Table 13-28. FlashEraseRegion Command Packet Format (Example)

FlashEraseRegion	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x0C 0x00
	crc16	0xF9 0x A6
Command packet	commandTag	0x02, kCommandTag_FlashEraseRegion
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	startAddress	0x00 0x00 0x00 0x00 (0x0000_0000)
	byte count	0x00 0x04 0x00 0x00 (0x400)

The FlashEraseRegion command has no data phase.

**Response:** The target (Kinetis Bootloader ) will return a GenericResponse packet with one of following error status codes.

Table 13-29. FlashEraseRegion Response Status Codes

Status Code
kStatus_Success (0x0)
kStatus_MemoryRangeInvalid (0x10200)
kStatus_FlashAlignmentError (0x101)
kStatus_FlashAddressError (0x102)
kStatus_FlashAccessError (0x103)
kStatus_FlashProtectionViolation (0x104)
kStatus_FlashCommandFailure (0x105)

### 13.3.8.8 FlashEraseAllUnsecure command

The FlashEraseAllUnsecure command performs a mass erase of the flash memory, including protected sectors. Flash security is immediately disabled if it (flash security) was enabled, and the FSEC byte in the flash configuration field at address 0x40C is programmed to 0xFE. However, if the mass erase enable option in the FSEC field is disabled, then the FlashEraseAllUnsecure command will fail.

The FlashEraseAllUnsecure command requires no parameters.

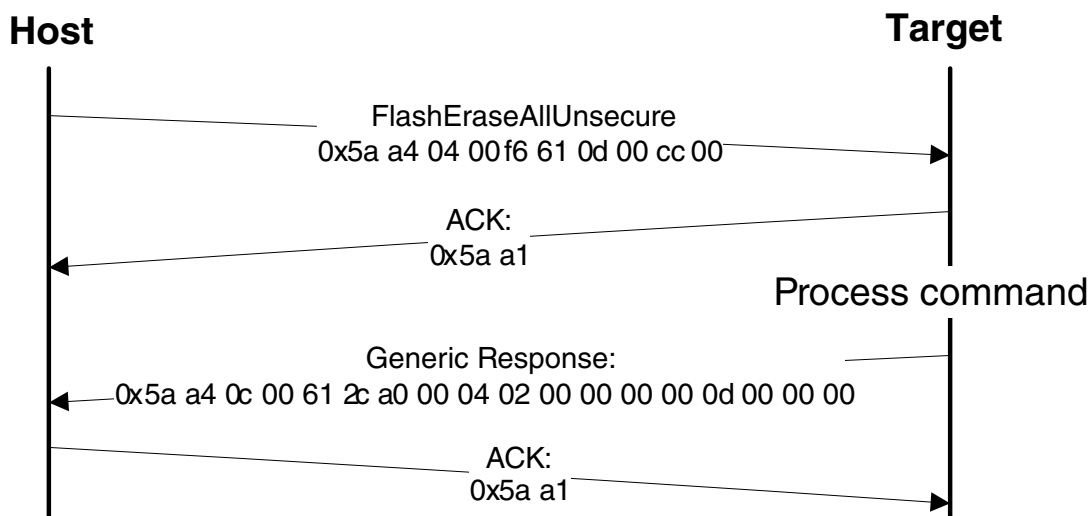


Figure 13-11. Protocol Sequence for FlashEraseAll Command

Table 13-30. FlashEraseAllUnsecure Command Packet Format (Example)

FlashEraseAllUnsecure	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x04 0x00
	crc16	0xF6 0x61
Command packet	commandTag	0x0D - FlashEraseAllUnsecure
	flags	0x00
	reserved	0x00
	parameterCount	0x00

The FlashEraseAllUnsecure command has no data phase.

**Response:** The target (Kinetis Bootloader) 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.

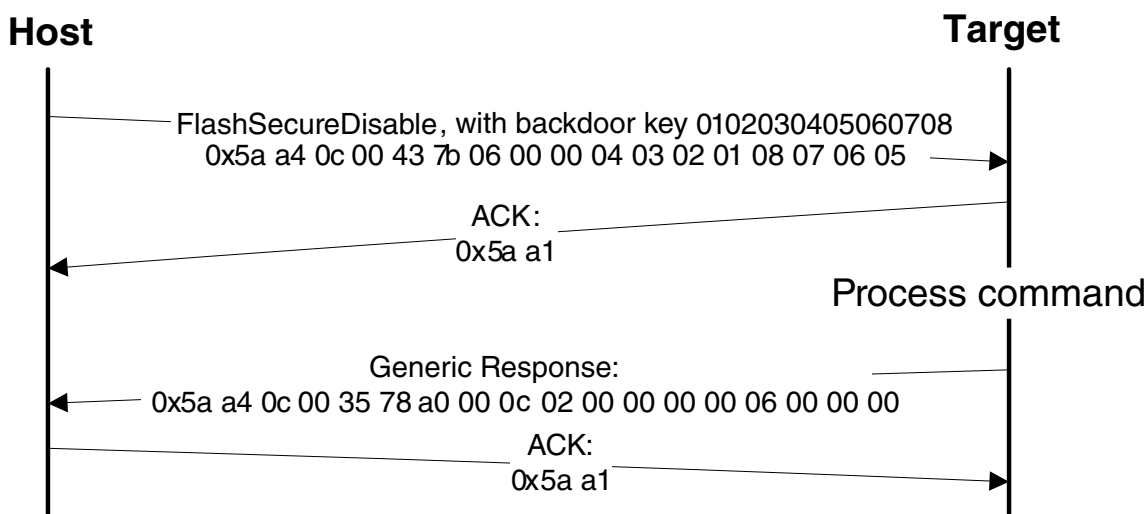
### 13.3.8.9 FlashSecurityDisable command

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 13-12. 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

Table continues on the next page...

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

FlashSecurityDisable	Parameter	Value
	crc16	0x43 0x7B
Command packet	commandTag	0x06 - FlashSecurityDisable
	flags	0x00
	reserved	0x00
	parameterCount	0x02
	Backdoorkey_low	0x04 0x03 0x02 0x01
	Backdoorkey_high	0x08 0x07 0x06 0x05

The FlashSecurityDisable command has no data phase.

**Response:** The target (Kinetis Bootloader) 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.

### 13.3.8.10 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, FlashEraseRegion, or FlashEraseAllUnsecure command.
- Writing to flash requires the start address to be 4-byte aligned ([1:0] = 00).
- The byte count will be rounded up to a multiple of 4, 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 13-33. Parameters for WriteMemory Command**

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

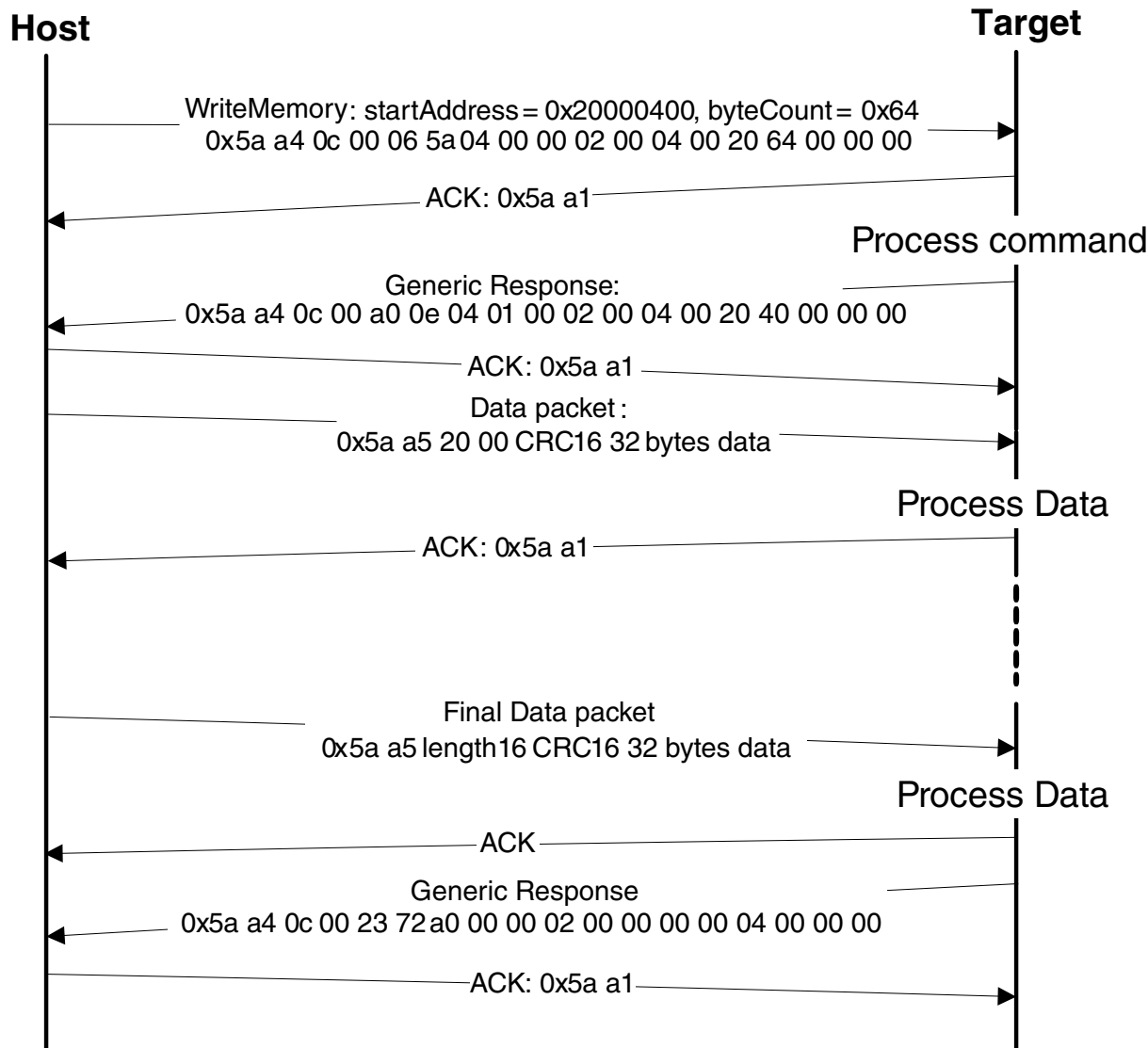


Figure 13-13. Protocol Sequence for WriteMemory Command

Table 13-34. 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 Bootloader ) 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.

### 13.3.9 Bootloader Exit state

The Kinetis Bootloader 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.

## 13.4 Peripherals Supported

This section describes the peripherals supported by the Kinetis ROM Bootloader. To use an interface for bootloader communications, the peripheral must be enabled in the BCA, as shown in [Table 13-3](#). If the BCA is invalid (such as all 0xFF bytes), then all peripherals will be enabled by default.

### 13.4.1 I2C Peripheral

The Kinetis Bootloader in KLx3 ROM 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.

Customizing an I2C slave address is also supported. This feature is enabled if the Bootloader Configuration Area (BCA) (shown in [Table 13-3](#)) is enabled (tag field is filled with 'kcfg') and the i2cSlaveAddress field is filled with a value other than 0xFF. Otherwise, 0x10 is used as the default I2C slave address.

The maximum supported I2C baud rate depends on corresponding clock configuration field in the BCA. Typical supported baud rate is 400 kbps with factory settings. Actual supported baud rate may be lower or higher than 400 kbps, depending on the actual value of the clockFlags and the clockDivider fields.

Because the I2C peripheral serves as an I2C slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

- An incoming packet is sent by the host with a selected I2C slave address and the direction bit is set as write.
- An outgoing packet is read by the host with a selected I2C slave address and the direction bit is set as read.
- 0x00 will be sent as the response to host if the target is busy with processing or preparing data.

The following flow charts demonstrate the communication flow of how the host reads ping packet, ACK and response from the target.

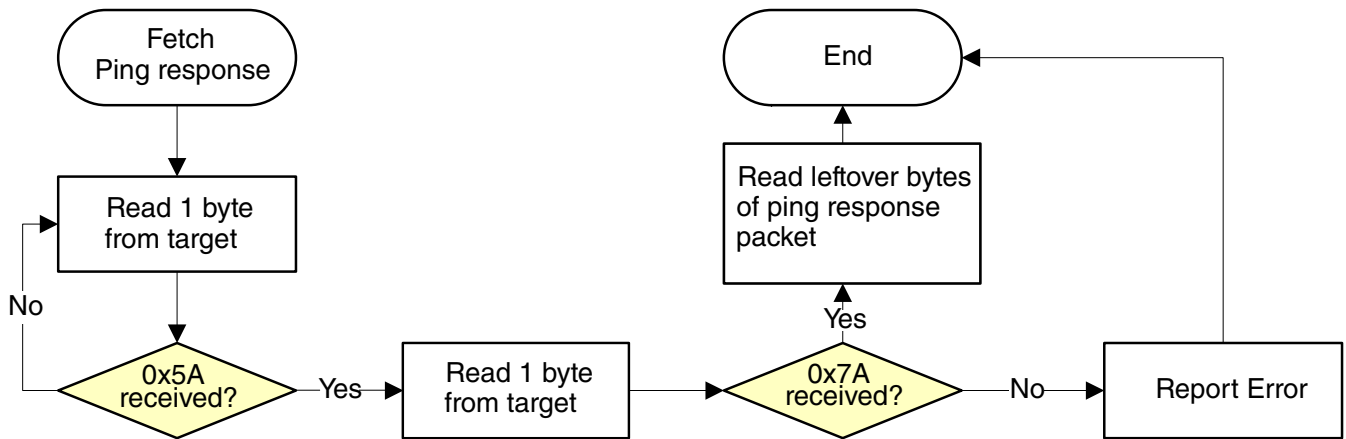


Figure 13-14. Host reads ping response from target via I2C

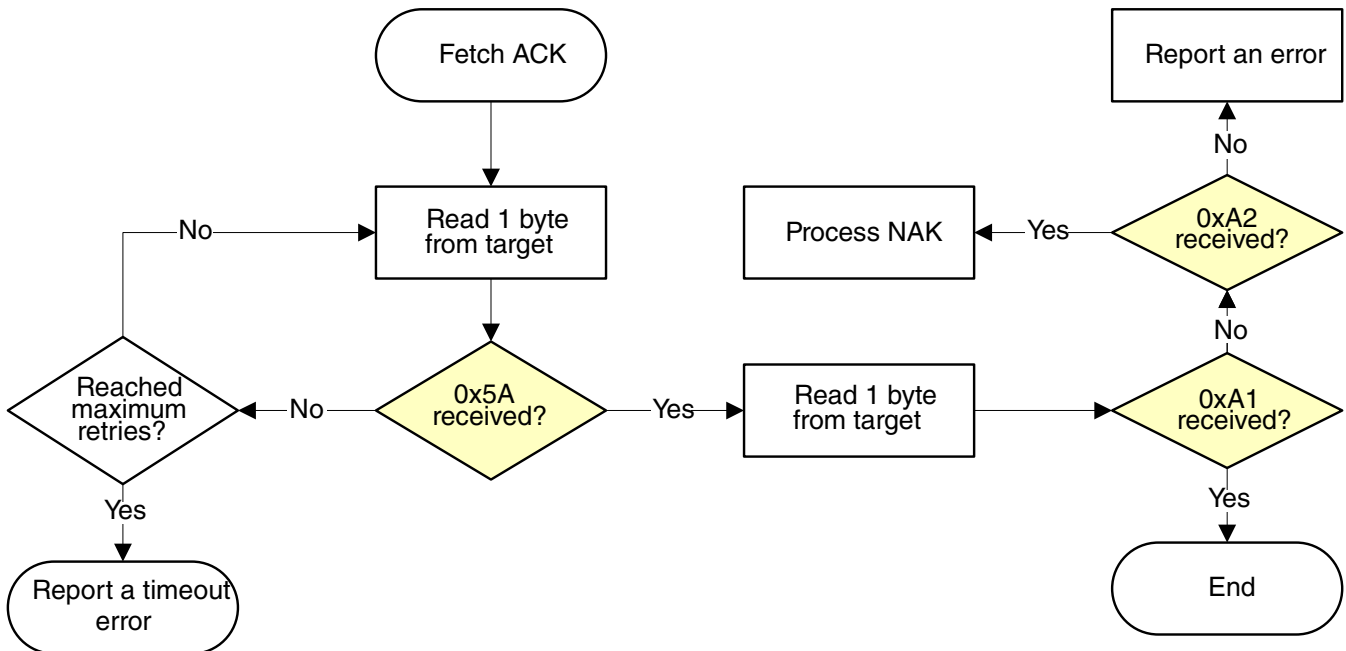


Figure 13-15. Host reads ACK packet from target via I2C



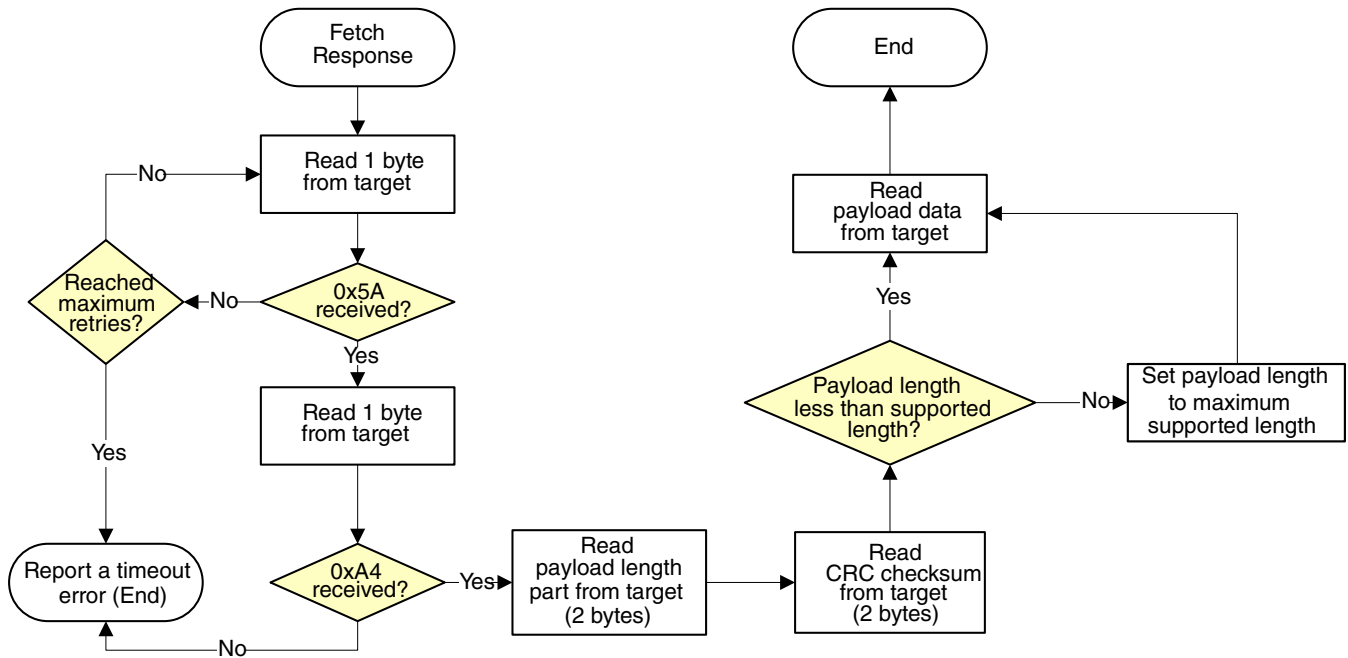


Figure 13-16. Host reads response from target via I2C

## 13.4.2 SPI Peripheral

The Kinetis Bootloader in KLx3 ROM supports loading data into flash via the SPI peripheral, where the SPI peripheral serves as a SPI slave.

Maximum supported baud rate of SPI depends on the clock configuration fields in the Bootloader Configuration Area (BCA) shown in [Table 13-3](#). The typical supported baud rate is 400 kbps with the factory settings. The actual baud rate is lower or higher than 400 kbps, depending on the actual value of the clockFlags and clockDivider fields in the BCA.

The SPI peripheral uses the following bus attributes:

- Clock Phase = 1 (Second Edge)
- Clock Polarity = 1 (Active Low)

Because the SPI peripheral in KLx3 ROM 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:

## Peripherals Supported

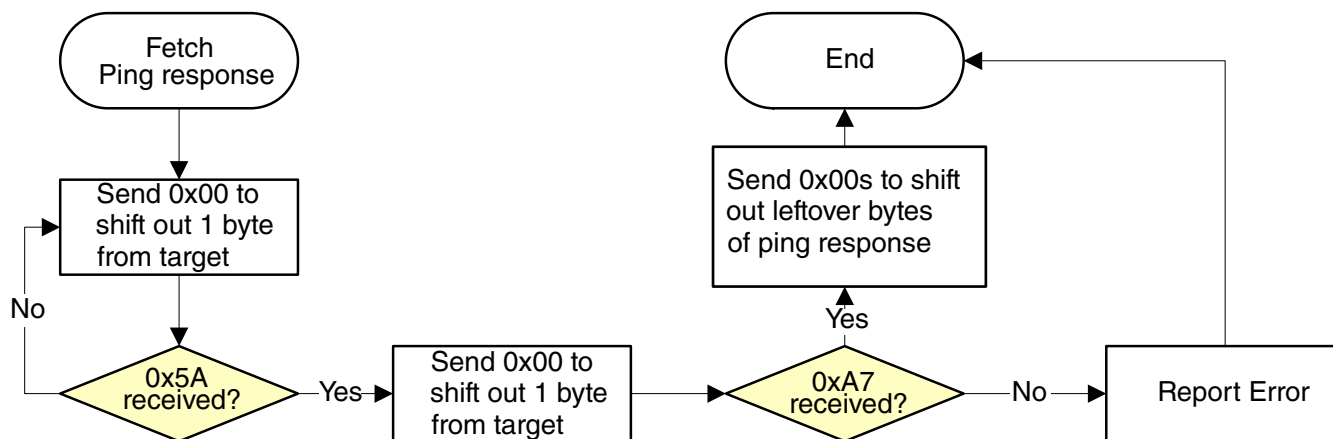
- Processing incoming packet
- Preparing outgoing data
- Received invalid data

The SPI bus configuration is:

- Phase = 1; data is sampled on rising edges
- Polarity = 1; idle is high
- MSB is transmitted first

For any transfer where the target does not have actual data to send, the target (slave) is responsible for ensuring that 0x00 bytes will be returned to the host (master). The host uses framing packets to identify real data and not "dummy" 0x00 bytes (which do not have framing packets).

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



**Figure 13-17. Host reads ping packet from target via SPI**

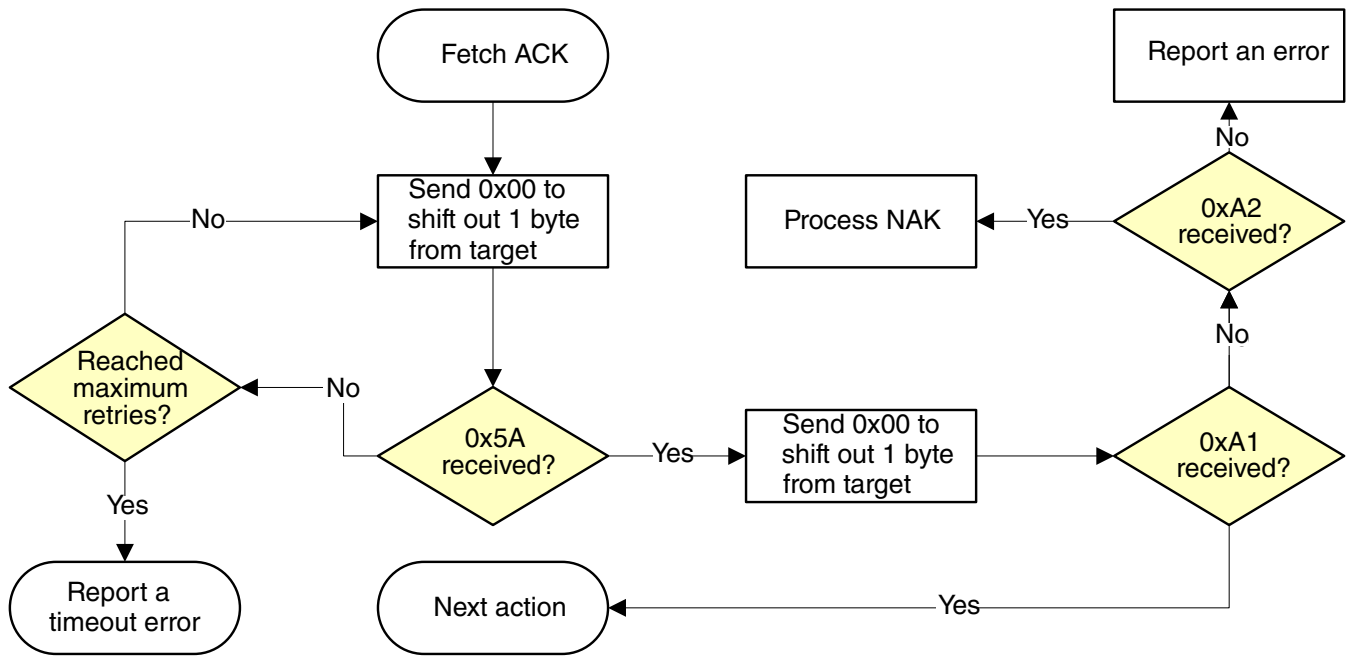


Figure 13-18. Host reads ACK from target via SPI

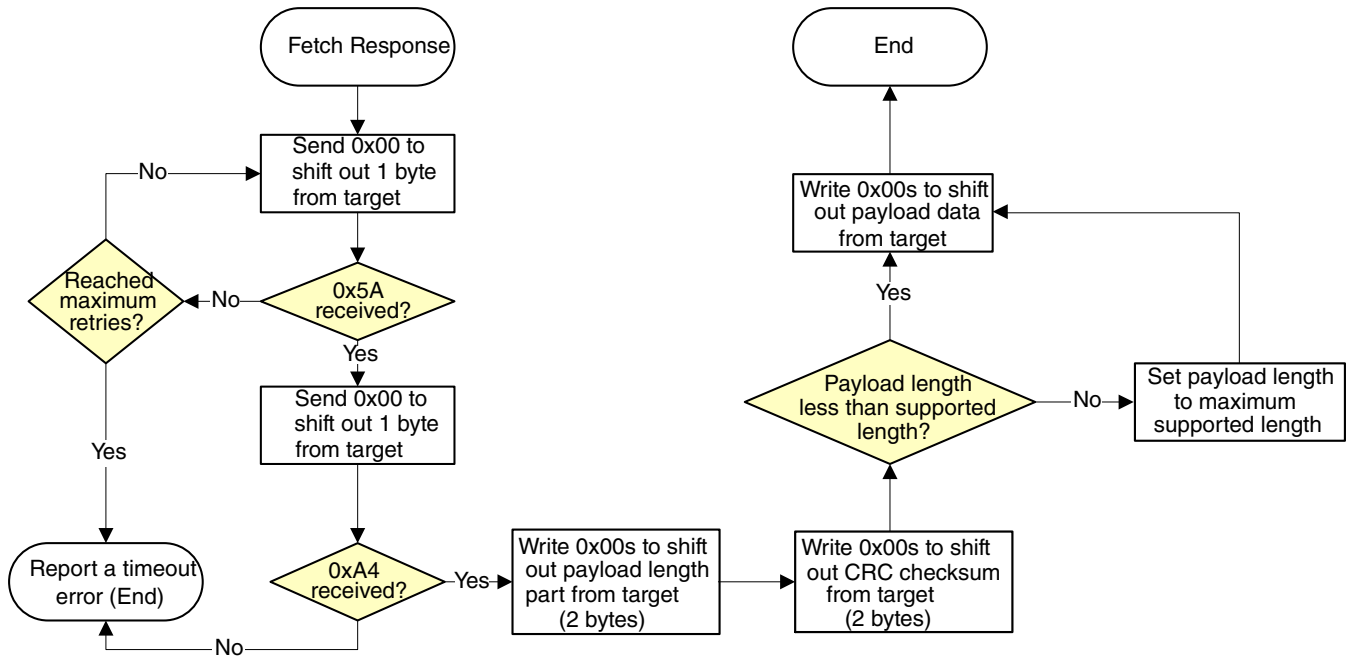


Figure 13-19. Host reads response from target via SPI

### 13.4.3 LPUART Peripheral

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

**Autobaud feature:** If LPUART $n$  is used to connect to the bootloader, then the LPUART $n$ \_RX pin must be kept high and not left floating during the detection phase in order to comply with the autobaud detection algorithm. After the bootloader detects the ping packet (0x5A 0xA6) on LPUART $n$ \_RX, the bootloader firmware executes the autobaud sequence. If the baudrate is successfully detected, then the bootloader 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 Bootloader then enters a loop, waiting for bootloader commands via the LPUART peripheral.

### NOTE

The data bytes of the ping packet must be sent continuously (with no more than 80 ms between bytes) in a fixed LPUART transmission mode (8-bit data, no parity bit and 1 stop bit). If the bytes of the ping packet are sent one-by-one with more than 80 ms delay between them, then the autobaud detection algorithm may calculate an incorrect baud rate. In this case, the autobaud detection state machine should be reset.

**Supported baud rates:** The baud rate is closely related to the MCU core and system clock frequencies. Typical baud rates supported are 9600, 19200, 38400, 57600, and 115200. Of course, to influence the performance of autobaud detection, the clock configuration in BCA can be changed.

**Packet transfer:** After autobaud detection succeeds, bootloader communications can take place over the LPUART 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

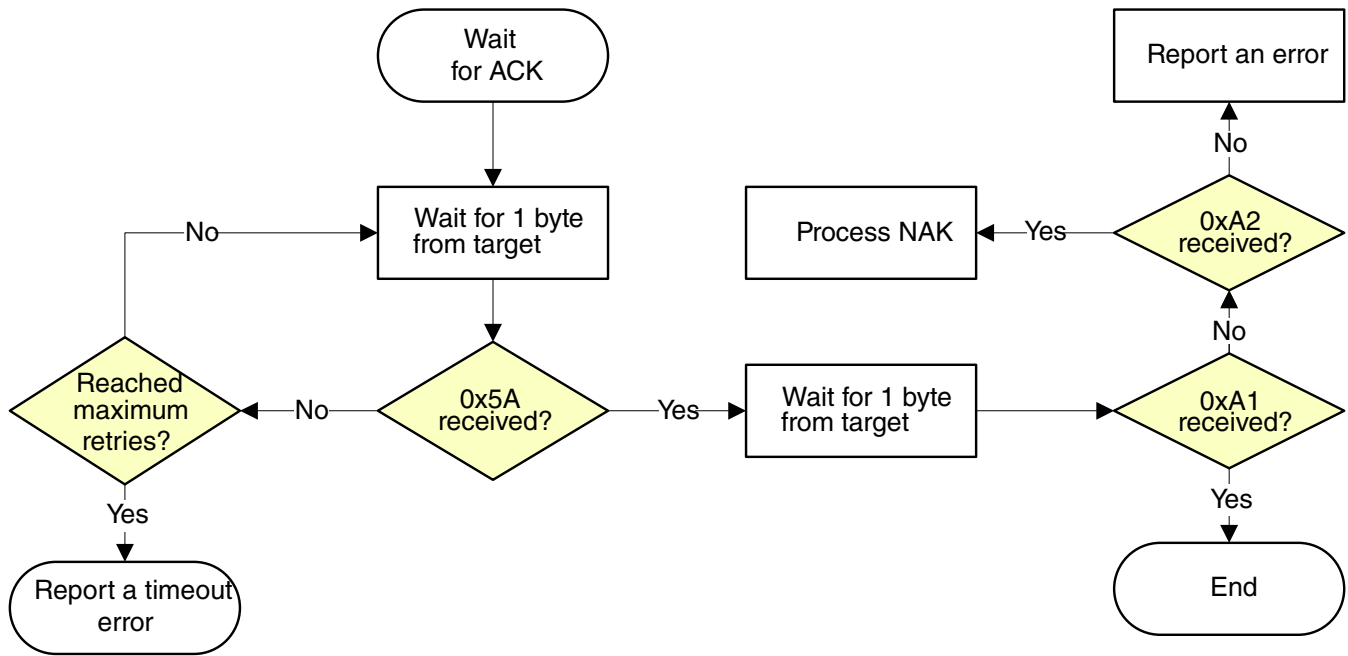


Figure 13-20. Host reads an ACK from target via LPUART

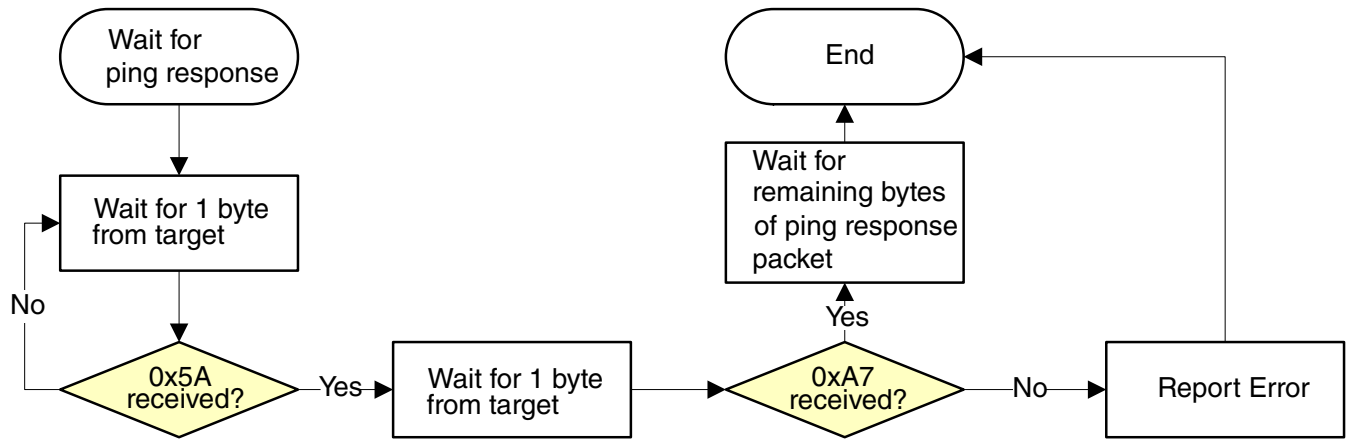


Figure 13-21. Host reads a ping response from target via LPUART

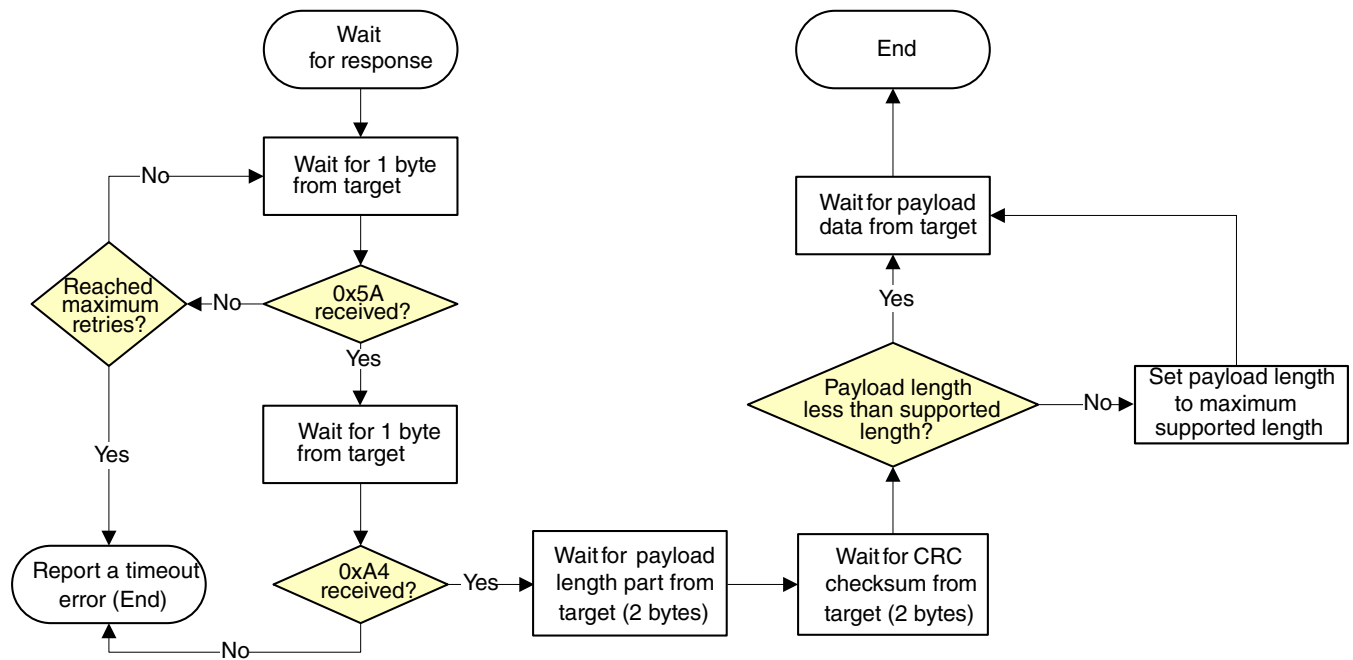


Figure 13-22. Host reads a command response from target via LPUART

## 13.5 Get/SetProperty Command Properties

This section lists the properties of the GetProperty and SetProperty commands.

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 bootloader 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 bootloader.
VerifyWrites	Yes	0Ah	4	Controls whether the bootloader 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 bootloader. 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>
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

## 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 bootloader.

**Table 13-36. Fields of CurrentVersion property:**

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

### 13.5.1.2 AvailablePeripherals Property

The value of this property is a bitfield that lists the peripherals supported by the bootloader 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	USB DFU	USB CDC	USB HID	CAN Slave	SPI Slave	I2C Slave	LPUART

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 bootloader. 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:

$$\text{mask} = 1 \ll (\text{tag} - 1)$$

**Table 13-38. Command bits:**

Bit	[31:17]	[16]	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Command	Reserved	Reserved	Reserved	Reserved	Reserved	FlashEraseAllUnsecure	SetProperty	Reset	Call	Execute	ReceiveSBFile	GetProperty	FlashSecurityDisable	FillMemory	WriteMemory	ReadMemory	FlashEraseRegion	FlashEraseAll

## 13.6 Kinetis Bootloader Status Error Codes

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

**Table 13-39. Kinetis Bootloader Status Error Codes, sorted by Value**

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.

*Table continues on the next page...*



**Table 13-39. Kinetis Bootloader Status Error Codes, sorted by Value (continued)**

Error Code	Value	Description
kStatus_Fail	1	Operation failed with a generic error.
kStatus_ReadOnly	2	Requested value cannot be changed because it is read-only.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_FlashSizeError	100	Not used.
kStatus_FlashAlignmentError	101	Address or length does not meet required alignment.
kStatus_FlashAddressError	102	Address or length is outside addressable memory.
kStatus_FlashAccessError	103	The FTFA_FSTAT[ACCERR] bit is set.
kStatus_FlashProtectionViolation	104	The FTFA_FSTAT[FPVIOL] bit is set.
kStatus_FlashCommandFailure	105	The FTFA_FSTAT[MGSTAT0] bit is set.
kStatus_FlashUnknownProperty	106	Unknown Flash property.
kStatus_FlashEraseKeyError	107	The key provided does not match the programmed flash key.
kStatus_FlashRegionExecuteOnly	108	The area of flash is protected as execute only.
kStatus_I2C_SlaveTxUnderrun	200	I2C Slave TX Underrun error.
kStatus_I2C_SlaveRxOverrun	201	I2C Slave RX Overrun error.
kStatus_I2C_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.
kStatus_Ping	10003	Internal: received ping during command phase.
kStatusRomLdrSectionOverrun	10100	The loader has finished processing the SB file.
kStatusRomLdrSignature	10101	The signature of the SB file is incorrect.
kStatusRomLdrSectionLength	10102	The section length in chunks is invalid.
kStatusRomLdrUnencryptedOnly	10103	An encrypted SB file has been sent and decryption support is not available.
kStatusRomLdrEOFReached	10104	The end of the SB file has been reached.
kStatusRomLdrChecksum	10105	The checksum of a command tag block is invalid.
kStatusRomLdrCrc32Error	10106	The CRC-32 of the data for a load command is incorrect.
kStatusRomLdrUnknownCommand	10107	An unknown command was found in the SB file.
kStatusRomLdrIdNotFound	10108	There was no bootable section found in the SB file.
kStatusRomLdrDataUnderrun	10109	The SB state machine is waiting for more data.
kStatusRomLdrJumpReturned	10110	The function that was jumped to by the SB file has returned.
kStatusRomLdrCallFailed	10111	The call command in the SB file failed.

*Table continues on the next page...*

**Table 13-39. Kinetis Bootloader Status Error Codes, sorted by Value (continued)**

Error Code	Value	Description
kStatusRomLdrKeyNotFound	10112	A matching key was not found in the SB file's key dictionary to unencrypt the section.
kStatusRomLdrSecureOnly	10113	The SB file sent is unencrypted and security on the target is enabled.
kStatusMemoryRangeInvalid	10200	Memory range conflicts with a protected region.
kStatus_UnknownProperty	10300	The requested property value is undefined.
kStatus_ReadOnlyProperty	10301	The requested property value cannot be written.
kStatus_InvalidPropertyValue	10302	The specified property value is invalid.
kStatus_AppCrcCheckPassed	10400	CRC check is valid and passed.
kStatus_AppCrcCheckFailed	10401	CRC check is valid but failed.
kStatus_AppCrcCheckInactive	10402	CRC check is inactive.
kStatus_AppCrcCheckInvalid	10403	CRC check is invalid, because the BCA is invalid or the CRC parameters are unset (all 0xFF bytes).
kStatus_AppCrcCheckOutOfRange	10404	CRC check is valid but addresses are out of range.

# Chapter 14

## System Mode Controller (SMC)

### 14.1 Chip-specific SMC information

This device does not support VLLS2 power mode. Ignore the VLLS2 in the following sections.

### 14.2 Introduction

The System Mode Controller (SMC) is responsible for sequencing the system into and out of all low-power Stop and Run modes.

Specifically, it monitors events to trigger transitions between power modes while controlling the power, clocks, and memories of the system to achieve the power consumption and functionality of that mode.

This chapter describes all the available low-power modes, the sequence followed to enter/exit each mode, and the functionality available while in each of the modes.

The SMC is able to function during even the deepest low power modes.

See [AN4503: Power Management for Kinetis MCUs](#) for further details on using the SMC.

### 14.3 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.

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 14-1. Power modes**

Mode	Description
RUN	The MCU can be run at full speed and the internal supply is fully regulated, that is, in run regulation. This mode is also referred to as Normal Run mode.
WAIT	The core clock is gated off. The system clock continues to operate. Bus clocks, if enabled, continue to operate. Run regulation is maintained.
STOP	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid.
VLPR	The core, system, bus, and flash clock maximum frequencies are restricted in this mode. See the Power Management chapter for details about the maximum allowable frequencies.

*Table continues on the next page...*

**Table 14-1. 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.
LLS	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.
VLLS3	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic. All system RAM contents are retained and I/O states are held. Internal logic states are not retained.
VLLS1	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic and all system RAM. I/O states are held. Internal logic states are not retained.
VLLS0	The core clock is gated off. System clocks to other masters and bus clocks are gated off after all stop acknowledge signals from supporting peripherals are valid. The MCU is placed in a low leakage mode by powering down the internal logic and all system RAM. I/O states are held. Internal logic states are not retained. The 1kHz LPO clock is disabled and the power on reset (POR) circuit can be optionally enabled using STOPCTRL[PORPO].

## 14.4 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	See section	14.4.1/214
4007_E001	Power Mode Control register (SMC_PMCTRL)	8	R/W	See section	14.4.2/215
4007_E002	Stop Control Register (SMC_STOPCTRL)	8	R/W	03h	14.4.3/217
4007_E003	Power Mode Status register (SMC_PMSTAT)	8	R	See section	14.4.4/218

**14.4.1 Power Mode Protection register (SMC\_PMPROT)**

This register provides protection for entry into any low-power run or stop mode. The enabling of the low-power run or stop mode occurs by configuring the Power Mode Control register (PMCTRL).

The PMPROT register can be written only once after any system reset.

If the MCU is configured for a disallowed or reserved power mode, the MCU remains in its current power mode. For example, if the MCU is in normal RUN mode and AVLP is 0, an attempt to enter VLPR mode using PMCTRL[RUNM] is blocked and PMCTRL[RUNM] remains 00b, indicating the MCU is still in Normal Run mode.

**NOTE**

This register is reset on Chip Reset not VLLS and by reset types that trigger Chip Reset not VLLS. It is unaffected by reset types that do not trigger Chip Reset not VLLS. See the Reset section details for more information.

Address: 4007\_E000h base + 0h offset = 4007\_E000h

Bit	7	6	5	4	3	2	1	0
Read	0	0	AVLP	0	ALLS	0	AVLLS	0
Write								
Reset	0	0	*	0	0	0	0	0

\* Notes:

- AVLP field: When booting in run mode, the reset value is 0. When booting in VLPR mode, the reset value is 1.

**SMC\_PMPROT field descriptions**

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

*Table continues on the next page...*

**SMC\_PMPROT field descriptions (continued)**

Field	Description
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 LLS is not allowed 1 LLS 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>

**14.4.2 Power Mode Control register (SMC\_PMCTRL)**

The PMCTRL register controls entry into low-power Run and Stop modes, provided that the selected power mode is allowed via an appropriate setting of the protection (PMPROT) register.

**NOTE**

This register is reset on Chip POR not VLLS and by reset types that trigger Chip POR not VLLS. It is unaffected by reset types that do not trigger Chip POR not VLLS. See the Reset section details for more information.

Address: 4007\_E000h base + 1h offset = 4007\_E001h

Bit	7	6	5	4	3	2	1	0
Read	Reserved		RUNM		0	STOPA		STOPM
Write	Reserved		RUNM		0	STOPA		STOPM
Reset	0	*	*	0	0	0	0	0

## Memory map and register descriptions

\* Notes:

- RUNM field: When booting in run mode, the reset value is 00. When booting in VLPR mode, the reset value is 01.

### 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.  00 Normal Run mode (RUN) 01 Reserved 10 Very-Low-Power Run mode (VLPR) 11 Reserved
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 STOPA	Stop Aborted  When set, this read-only status bit indicates an interrupt 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.
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, the VLLSM field in the STOPCTRL register is used to further select the particular VLLS 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) 001 Reserved 010 Very-Low-Power Stop (VLPS) 011 Low-Leakage Stop (LLS) 100 Very-Low-Leakage Stop (VLLSx) 101 Reserved 110 Reserved 111 Reserved



### 14.4.3 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	LPOPO	VLLSM		
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 (or VLPR) mode, the PMC, MCG and flash remain fully powered, allowing the device to wakeup almost instantaneously at the expense of higher power consumption. In PSTOP2, only system clocks are gated allowing peripherals running on bus clock to remain fully functional. In PSTOP1, both system and bus clocks are gated.</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> <p>0 POR detect circuit is enabled in VLLS0            1 POR detect circuit is disabled in VLLS0</p>
4 Reserved	<p>This field is reserved.            This read-only field is reserved and always has the value 0.</p>
3 LPOPO	<p>LPO Power Option</p> <p>Controls whether the 1 kHz LPO clock is enabled in LLS/VLLSx modes.</p> <p><b>NOTE:</b> During VLLS0 mode, the LPO clock is disabled by hardware and this bit has no effect.</p> <p>0 LPO clock is enabled in LLS/VLLSx            1 LPO clock is disabled in LLS/VLLSx</p>
VLLSM	VLLS Mode Control

Table continues on the next page...

**SMC\_STOPCTRL field descriptions (continued)**

Field	Description
	This field controls which VLLS sub-mode to enter if STOPM = VLLSx.
000	VLLS0
001	VLLS1
010	Reserved
011	VLLS3
100	Reserved
101	Reserved
110	Reserved
111	Reserved

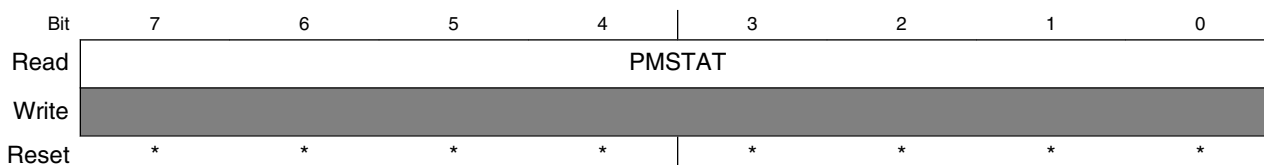
**14.4.4 Power Mode Status register (SMC\_PMSTAT)**

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



\* Notes:

- PMSTAT field: When booting in run mode, the reset value is 0x00. When booting in VLPR mode, the reset value is 0x04.

**SMC\_PMSTAT field descriptions**

Field	Description
PMSTAT	<p>Power Mode Status</p> <p><b>NOTE:</b> When debug is enabled, the PMSTAT will not update to STOP or VLPS</p> <p><b>NOTE:</b> When a PSTOP mode is enabled, the PMSTAT will not update to STOP or VLPS</p> <p><b>NOTE:</b> Since the RUNM bits in the PMCTRL register are reset to VLPR on any Chip Reset not VLLS, the PMSTAT will update to VLPR shortly after the reset sequence is complete.</p> <p>0000_0001 Current power mode is RUN.</p> <p>0000_0010 Current power mode is STOP.</p>

**SMC\_PMSTAT field descriptions (continued)**

Field	Description
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	Reserved

## 14.5 Functional description

### 14.5.1 Power mode transitions

The following figure shows the power mode state transitions available on the chip. Any reset will initially bring the MCU back to the normal RUN state. However, in order to minimize peak power consumption, the RUNM bits in the PMCTRL register can be reset to VLPR via Flash IFR settings, causing the SMC to begin transitioning the MCU into VLPR mode during the reset recovery sequence.

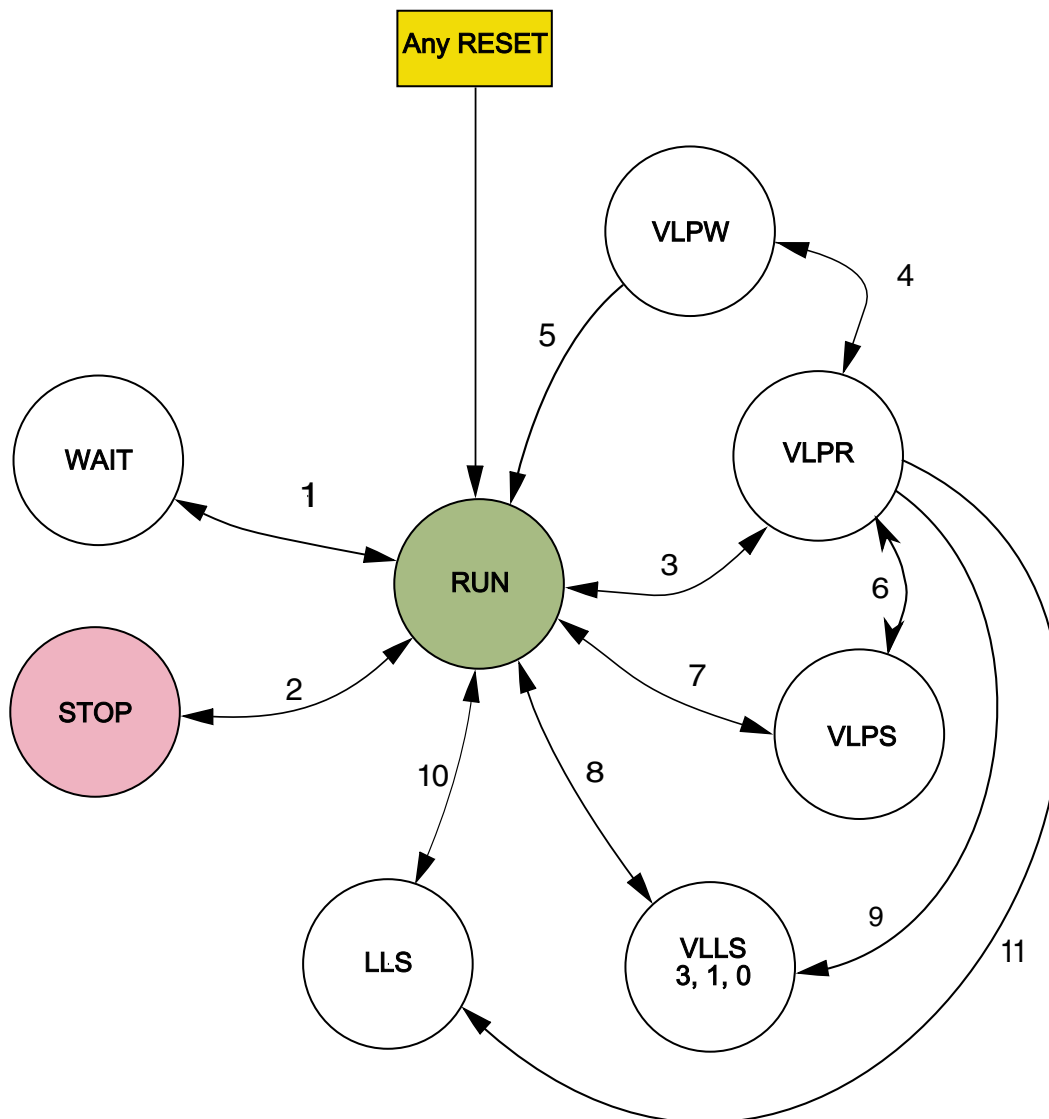


Figure 14-1. Power mode state diagram

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

Table 14-2. Power mode transition triggers

Transition #	From	To	Trigger conditions
1	RUN	WAIT	Sleep-now or sleep-on-exit modes entered with SLEEPDEEP clear, controlled in System Control Register in ARM core. See note. <sup>1</sup>
	WAIT	RUN	Interrupt or Reset
2	RUN	STOP	PMCTRL[RUNM]=00, PMCTRL[STOPM]=000 <sup>2</sup>

Table continues on the next page...

Table 14-2. Power mode transition triggers (continued)

Transition #	From	To	Trigger conditions
			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. <b>NOTE:</b> In order to limit peak current, PMPROT[AVLP] and PMCTRL[RUNM] bits can be set on any Reset via Flash IFR settings, causing the SMC to transition the MCU from RUN->VLPR during the reset recovery sequence.
	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

Table continues on the next page...

**Table 14-2. Power mode transition triggers (continued)**

Transition #	From	To	Trigger conditions
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.
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

1. If debug is enabled, the core clock remains to support debug.
2. If PMCTRL[STOPM]=000 and STOPCTRL[PSTOPO]=01 or 10, then only a Partial Stop mode is entered instead of STOP
3. If PMCTRL[STOPM]=000 and STOPCTRL[PSTOPO]=00, then VLPS mode is entered instead of STOP. If PMCTRL[STOPM]=000 and STOPCTRL[PSTOPO]=01 or 10, then only a Partial Stop mode is entered instead of VLPS

## 14.5.2 Power mode entry/exit sequencing

When entering or exiting low-power modes, the system must conform to an orderly sequence to manage transitions safely.

The SMC manages the system's entry into and exit from all power modes. This diagram illustrates the connections of the SMC with other system components in the chip that are necessary to sequence the system through all power modes.

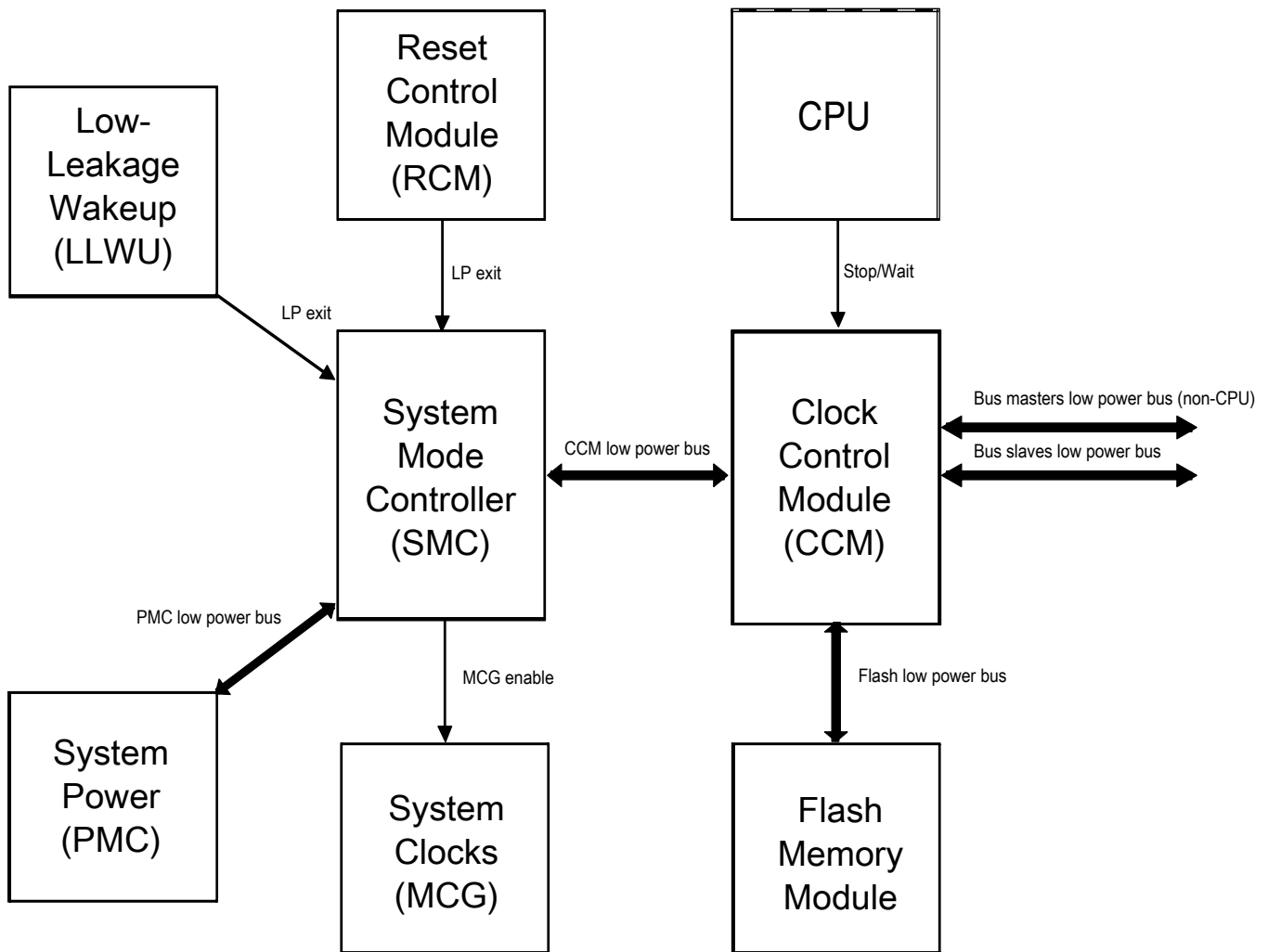


Figure 14-2. Low-power system components and connections

### 14.5.2.1 Stop mode entry sequence

Entry into a low-power stop mode (Stop, VLPS, LLS, VLLSx) is initiated by a CPU executing the WFI instruction. After the instruction is executed, the following sequence occurs:

1. The CPU clock is gated off immediately.
2. Requests are made to all non-CPU bus masters to enter Stop mode.
3. After all masters have acknowledged they are ready to enter Stop mode, requests are made to all bus slaves to enter Stop mode.
4. After all slaves have acknowledged they are ready to enter Stop mode, all system and bus clocks are gated off.
5. Clock generators are disabled in the MCG.
6. The on-chip regulator in the PMC and internal power switches are configured to meet the power consumption goals for the targeted low-power mode.

### 14.5.2.2 Stop mode exit sequence

Exit from a low-power stop mode is initiated either by a reset or an interrupt event. The following sequence then executes to restore the system to a run mode (RUN or VLPR):

1. The on-chip regulator in the PMC and internal power switches are restored.
2. Clock generators are enabled in the MCG.
3. System and bus clocks are enabled to all masters and slaves.
4. The CPU clock is enabled and the CPU begins servicing the reset or interrupt that initiated the exit from the low-power stop mode.

### 14.5.2.3 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.

### 14.5.2.4 Transition to wait modes

For wait modes (WAIT and VLPW), the CPU clock is gated off while all other clocking continues, as in RUN and VLPR mode operation. Some modules that support stop-in-wait functionality have their clocks disabled in these configurations.

### 14.5.2.5 Transition from stop modes to Debug mode

The debugger module supports a transition from STOP, WAIT, VLPS, and VLPW back to a Halted state when the debugger has been enabled. As part of this transition, system clocking is re-established and is equivalent to the normal RUN and VLPR mode clocking configuration.

## 14.5.3 Run modes

The run modes supported by this device can be found here.

- Run (RUN)
- Very Low-Power Run (VLPR)



### 14.5.3.1 RUN mode

This mode is selected after any reset. In order to reduce peak power consumption, however, the SMC will begin transitioning the MCU into VLPR mode during the reset recovery sequence. 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.

### 14.5.3.2 Very-Low Power Run (VLPR) mode

In VLPR mode, the on-chip voltage regulator is put into a stop mode regulation state. In this state, the regulator is designed to supply enough current to the MCU over a reduced frequency. To further reduce power in this mode, disable the clocks to unused modules using their corresponding clock gating control bits in the SIM's registers.

Before entering this mode, the following conditions must be met:

- The MCG must be configured in a mode which is supported during VLPR. See the Power Management details for information about these MCG modes.
- All clock monitors in the MCG must be disabled.
- The maximum frequencies of the system, bus, flash, and core are restricted. See the Power Management details about which frequencies are supported.
- Mode protection must be set to allow VLP modes, that is, PMPROT[AVLP] is 1.
- PMCTRL[RUNM] must be set to 10b to enter VLPR.
- Flash programming/erasing is not allowed.

#### NOTE

Do not increase the clock frequency while in VLPR mode, because the regulator is slow in responding and cannot manage fast load transitions. In addition, do not modify the clock source in the MCG module or any clock divider registers. Module clock enables in the SIM can be set, however should only be cleared one at a time to limit load transitions.

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 will initially cause the device to exit to RUN mode, however Flash IFR settings can be used to return the part to VLPR during the MCU reset flow.

## **14.5.4 Wait modes**

This device contains two different wait modes which are listed here.

- Wait
- Very-Low Power Wait (VLPW)

### **14.5.4.1 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, temporarily returning the device to RUN mode before transitioning into VLPR during the MCU reset flow.

### **14.5.4.2 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, temporarily returning the device to RUN mode before transitioning into VLPR during the MCU reset flow.

### 14.5.5 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)

#### 14.5.5.1 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, temporarily returning the device to RUN mode before transitioning into VLPR during the MCU reset flow.

### **14.5.5.2 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 cause an exit from VLPS mode, temporarily returning the device to RUN mode before transitioning into VLPR during the MCU reset flow.

### **14.5.5.3 Low-Leakage Stop (LLS) mode**

Low-Leakage Stop (LLS) mode 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 14-2](#).

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. If LLS was entered from VLPR mode, then the MCU will begin VLPR entry shortly after wake-up. 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, temporarily returning the device to normal RUN mode before transitioning into VLPR during the MCU reset flow. When LLS is exiting via the  $\overline{\text{RESET}}$  pin, RCM\_SRS0[PIN] and RCM\_SRS0[WAKEUP] are set.

#### 14.5.5.4 Very-Low-Leakage Stop (VLLSx) modes

This device contains these very low leakage modes:

- VLLS3
- VLLS1
- VLLS0

VLLSx is often used in this document to refer to all of these modes.

All VLLSx modes can be entered from normal RUN or VLPR modes.

The MCU enters the configured VLLS mode if:

- In Sleep-Now or Sleep-On-Exit mode, the SLEEPDEEP bit is set in the System Control Register in the ARM core, and
- The device is configured as shown in [Table 14-2](#).

In VLLS, the on-chip voltage regulator is in its stop-regulation state while most digital logic is powered off.

Before entering VLLS mode, the user should configure the Low-Leakage Wake-up (LLWU) module to enable the desired wakeup sources. The available wake-up sources in VLLS are detailed in the chip configuration details for this device.

After wakeup from VLLS, the device returns to normal RUN mode with a pending LLWU interrupt. If VLLS was entered from VLPR mode, then the MCU will begin VLPR entry shortly after wakeup. 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, temporarily returning the device to normal RUN mode before transitioning into VLPR during the MCU reset flow. When exiting VLLS via the  $\overline{\text{RESET}}$  pin, RCM\_SRS0[PIN] and RCM\_SRS0[WAKEUP] are set.

### 14.5.6 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 15

## Power Management Controller (PMC)

### 15.1 Introduction

The power management controller (PMC) contains the internal voltage regulator, power on reset (POR), low voltage detect system (LVD), and high voltage detect system (HVD).

See [AN4503: Power Management for Kinetis MCUs](#) for further details on using the PMC.

### 15.2 Features

A list of included PMC features can be found here.

- Internal voltage regulator
- Active POR providing brown-out detect
- Low-voltage detect supporting two low-voltage trip points with four warning levels per trip point

### 15.3 Low-voltage detect (LVD) system

This device includes a system to guard against low-voltage conditions. 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 ( $V_{LV\text{DH}}$ ) or low ( $V_{LV\text{DL}}$ ). The trip voltage is selected by `LV\text{DSC1}[LV\text{DV}]`. 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 (`LV\text{DSC1}[LV\text{DF}]`) operates in a level sensitive manner. `LV\text{DSC1}[LV\text{DF}]` is set

when the supply voltage falls below the selected trip point (VLVD).

LVDS1[LVDF] is cleared by writing 1 to LVDS1[LVDAK], but only if the internal supply has returned above the trip point; otherwise, LVDS1[LVDF] remains set.

- The Low Voltage Warning Flag (LVWF) in the Low Voltage Status and Control 2 Register (LVDS2[LVWF]) operates in a level sensitive manner. LVDS2[LVWF] is set when the supply voltage falls below the selected monitor trip point (VLVW). LVDS2[LVWF] is cleared by writing one to LVDS2[LVWAK], but only if the internal supply has returned above the trip point; otherwise, LVDS2[LVWF] remains set.

### 15.3.1 LVD reset operation

By setting LVDS1[LVDR], the LVD generates a reset upon detection of a low-voltage condition. The low-voltage detection threshold is determined by LVDS1[LVDV]. After an LVD reset occurs, the LVD system holds the MCU in reset until the supply voltage rises above this threshold. The LVD field in the SRS register of the RCM module (RCM\_SRS[LVD]) is set following an LVD or power-on reset.

### 15.3.2 LVD interrupt operation

By configuring the LVD circuit for interrupt operation (LVDS1[LVDR] set and LVDS1[LVDR] clear), LVDS1[LVDF] is set and an LVD interrupt request occurs upon detection of a low voltage condition. LVDS1[LVDF] is cleared by writing 1 to LVDS1[LVDAK].

### 15.3.3 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 LVDS2[LVWIE]. If enabled, an LVW interrupt request occurs when LVDS2[LVWF] is set. LVDS2[LVWF] is cleared by writing 1 to LVDS2[LVWAK].

LVDS2[LVWV] selects one of the four trip voltages:

- Highest:  $V_{LVW4}$

- Two mid-levels:  $V_{LVW3}$  and  $V_{LVW2}$
- Lowest:  $V_{LVW1}$

## 15.4 I/O retention

When in 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.

## 15.5 Memory map and register descriptions

Details about the PMC registers can be found here.

### NOTE

Different portions of PMC registers are reset only by particular reset types. Each register's description provides details. For more information about the types of reset on this chip, refer to the Reset section details.

The PMC registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

### PMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4007_D000	Low Voltage Detect Status And Control 1 register (PMC_LVDSC1)	8	R/W	10h	<a href="#">15.5.1/236</a>
4007_D001	Low Voltage Detect Status And Control 2 register (PMC_LVDSC2)	8	R/W	00h	<a href="#">15.5.2/237</a>
4007_D002	Regulator Status And Control register (PMC_REGSC)	8	R/W	04h	<a href="#">15.5.3/238</a>

### 15.5.1 Low Voltage Detect Status And Control 1 register (PMC\_LVDSC1)

This register contains status and control bits to support the low voltage detect function. This register should be written during the reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

While the device is in the very low power or low leakage modes, the LVD system is disabled regardless of LVDSC1 settings. To protect systems that must have LVD always on, configure the Power Mode Protection (PMPROT) register of the SMC module (SMC\_PMPROT) to disallow any very low power or low leakage modes from being enabled.

See the device's data sheet for the exact LVD trip voltages.

#### NOTE

The LVDV bits are reset solely on a POR Only event. The register's other bits are reset on Chip Reset Not VLLS. For more information about these reset types, refer to the Reset section details.

Address: 4007\_D000h base + 0h offset = 4007\_D000h

Bit	7	6	5	4	3	2	1	0
Read	LVDF	0	LVDIE	LVDRE	0		LVDV	
Write		LVDACK						
Reset	0	0	0	1	0	0	0	0

PMC\_LVDSC1 field descriptions

Field	Description
7 LVDF	Low-Voltage Detect Flag  This read-only status field indicates a low-voltage detect event.  0 Low-voltage event not detected 1 Low-voltage event detected
6 LVDACK	Low-Voltage Detect Acknowledge  This write-only field is used to acknowledge low voltage detection errors. Write 1 to clear LVDF. Reads always return 0.
5 LVDIE	Low-Voltage Detect Interrupt Enable  Enables hardware interrupt requests for LVDF.  0 Hardware interrupt disabled (use polling) 1 Request a hardware interrupt when LVDF = 1

Table continues on the next page...

### PMC\_LVDSC1 field descriptions (continued)

Field	Description
4 LVDRE	<p>Low-Voltage Detect Reset Enable</p> <p>This write-once bit enables LVDF events to generate a hardware reset. Additional writes are ignored.</p> <p>0 LVDF does not generate hardware resets 1 Force an MCU reset when LVDF = 1</p>
3–2 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
LVDV	<p>Low-Voltage Detect Voltage Select</p> <p>Selects the LVD trip point voltage (<math>V_{LVD}</math>).</p> <p>00 Low trip point selected (<math>V_{LVD} = V_{LVDL}</math>) 01 High trip point selected (<math>V_{LVD} = V_{LVDH}</math>) 10 Reserved 11 Reserved</p>

### 15.5.2 Low Voltage Detect Status And Control 2 register (PMC\_LVDSC2)

This register contains status and control bits to support the low voltage warning function.

While the device is in the very low power or low leakage modes, the LVD system is disabled regardless of LVDSC2 settings.

See the device's data sheet for the exact LVD trip voltages.

#### NOTE

The LVW trip voltages depend on LVWV and LVDV.

#### NOTE

LVWV is reset solely on a POR Only event. The other fields of the register are reset on Chip Reset Not VLLS. For more information about these reset types, refer to the Reset section details.

Address: 4007\_D000h base + 1h offset = 4007\_D001h

Bit	7	6	5	4	3	2	1	0
Read	LVWF	0	LVWIE	0		LVWV		
Write	LVWACK		LVWIE	0			LVWV	
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>
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 LVDS1[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>

### 15.5.3 Regulator Status And Control register (PMC\_REGSC)

The PMC contains an internal voltage regulator. The voltage regulator design uses a bandgap reference that is also available through a buffer as input to certain internal peripherals, such as the CMP and ADC. The internal regulator provides a status bit (REGONS) indicating the regulator is in run regulation.

#### NOTE

This register is reset on Chip Reset Not VLLS and by reset types that trigger Chip Reset not VLLS. See the Reset section details for more information.

Address: 4007\_D000h base + 2h offset = 4007\_D002h

Bit	7	6	5	4	3	2	1	0
Read	0	VLPO	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 VLPO	VLPx Option  When used in conjunction with BGEN, this bit allows additional clock sources and higher frequency operation (at the cost of higher power) to be selected during VLPx modes.  0 Operating frequencies and MCG clocking modes are restricted during VLPx modes as listed in the Power Management chapter. 1 If BGEN is also set, operating frequencies and MCG clocking modes are unrestricted during VLPx modes. Note that flash access frequency is still restricted however.
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 16

## Miscellaneous Control Module (MCM)

### 16.1 Introduction

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

#### 16.1.1 Features

The MCM includes the following features:

- Program-visible information on the platform configuration
- Crossbar master arbitration policy selection
- Flash controller speculation buffer and cache configurations

### 16.2 Memory map/register descriptions

The memory map and register descriptions found here describe the registers using byte addresses. The registers can be written only when in supervisor mode.

**MCM memory map**

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
F000_3008	Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)	16	R	0007h	<a href="#">16.2.1/242</a>
F000_300A	Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)	16	R	0005h	<a href="#">16.2.2/242</a>
F000_300C	Platform Control Register (MCM_PLACR)	32	R/W	0000_0050h	<a href="#">16.2.3/243</a>
F000_3040	Compute Operation Control Register (MCM_CPO)	32	R/W	0000_0000h	<a href="#">16.2.4/246</a>

### 16.2.1 Crossbar Switch (AXBS) Slave Configuration (MCM\_PLASC)

PLASC is a 16-bit read-only register identifying the presence/absence of bus slave connections to the device's crossbar switch.

Address: F000\_3000h base + 8h offset = F000\_3008h

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

#### MCM\_PLASC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ASC	Each bit in the ASC field indicates whether there is a corresponding connection to the crossbar switch's slave input port.  0 A bus slave connection to AXBS input port <i>n</i> is absent. 1 A bus slave connection to AXBS input port <i>n</i> is present.

### 16.2.2 Crossbar Switch (AXBS) Master Configuration (MCM\_PLAMC)

PLAMC is a 16-bit read-only register identifying the presence/absence of bus master connections to the device's crossbar switch.

Address: F000\_3000h base + Ah offset = F000\_300Ah

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

#### MCM\_PLAMC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
AMC	Each bit in the AMC field indicates whether there is a corresponding connection to the AXBS master input port.

*Table continues on the next page...*

**MCM\_PLAMC field descriptions (continued)**

Field	Description
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

**16.2.3 Platform Control Register (MCM\_PLACR)**

The PLACR register selects the arbitration policy for the crossbar masters and configures the flash memory controller.

The speculation buffer and cache in the flash memory controller is configurable via PLACR[15:10 ].

The speculation buffer is enabled only for instructions after reset. It is possible to have these states for the speculation buffer:

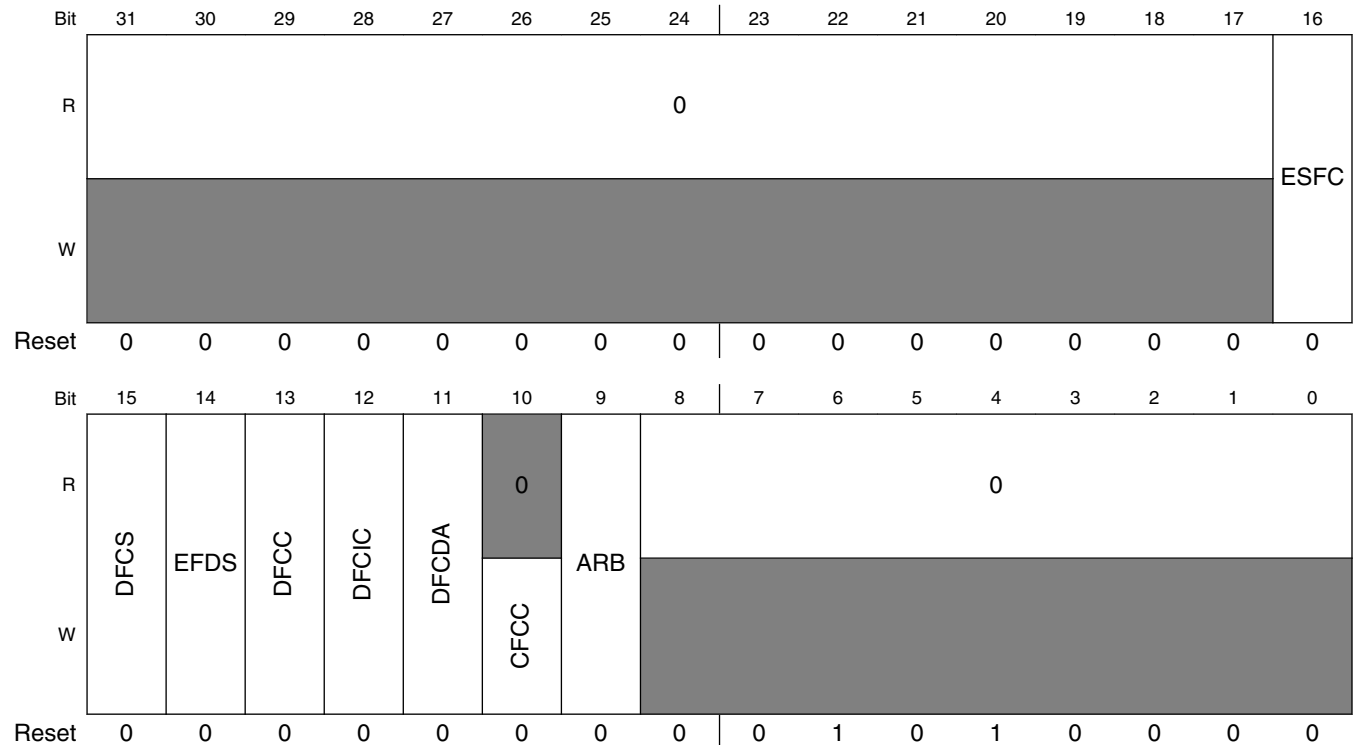
DFCS	EFDS	Description
0	0	Speculation buffer is on for instruction and off for data.
0	1	Speculation buffer is on for instruction and on for data.
1	X	Speculation buffer is off.

The cache in flash controller is enabled and caching both instruction and data type fetches after reset. It is possible to have these states for the cache:

DFCC	DFCIC	DFCDA	Description
0	0	0	Cache is on for both instruction and data.
0	0	1	Cache is on for instruction and off for data.
0	1	0	Cache is off for instruction and on for data.
0	1	1	Cache is off for both instruction and data.
1	X	X	Cache is off.

## Memory map/register descriptions

Address: F000\_3000h base + Ch offset = F000\_300Ch



### MCM\_PLACR field descriptions

Field	Description
31–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ESFC	<p>Enable Stalling Flash Controller</p> <p>Enables stalling flash controller when flash is busy.</p> <p>When software needs to access the flash memory while a flash memory resource is being manipulated by a flash command, software can enable a stall mechanism to avoid a read collision. The stall mechanism allows software to execute code from the same block on which flash operations are being performed. However, software must ensure the sector the flash operations are being performed on is not the same sector from which the code is executing.</p> <p>ESFC enables the stall mechanism. This bit must be set only just before the flash operation is executed and must be cleared when the operation completes.</p> <p>0 Disable stalling flash controller when flash is busy. 1 Enable stalling flash controller when flash is busy.</p>
15 DFCS	<p>Disable Flash Controller Speculation</p> <p>Disables flash controller speculation.</p> <p>0 Enable flash controller speculation. 1 Disable flash controller speculation.</p>
14 EFDS	<p>Enable Flash Data Speculation</p> <p>Enables flash data speculation.</p>

Table continues on the next page...

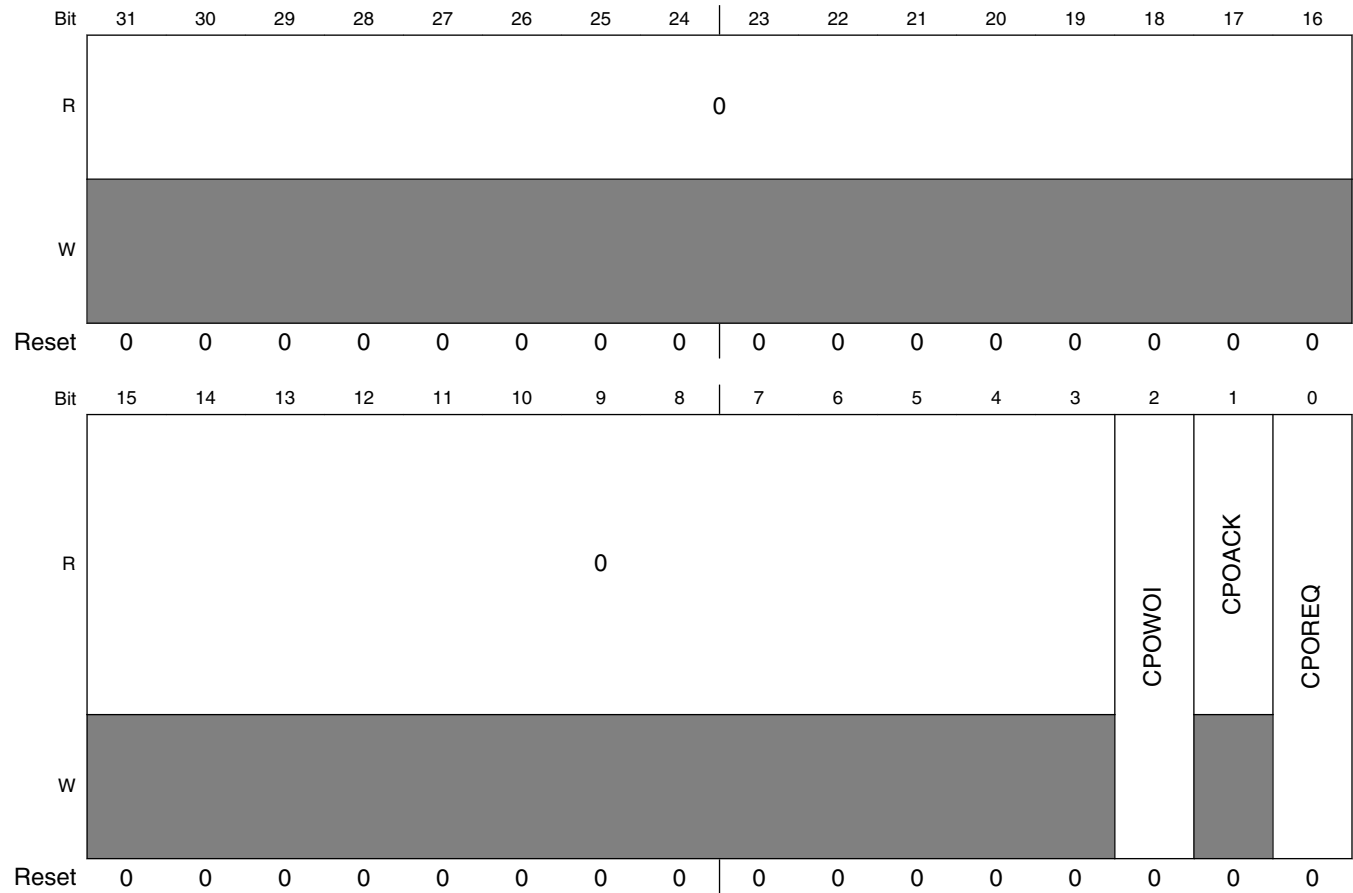
**MCM\_PLACR field descriptions (continued)**

<b>Field</b>	<b>Description</b>
	0 Disable flash data speculation. 1 Enable flash data speculation.
13 DFCC	Disable Flash Controller Cache Disables flash controller cache. 0 Enable flash controller cache. 1 Disable flash controller cache.
12 DFCIC	Disable Flash Controller Instruction Caching Disables flash controller instruction caching. 0 Enable flash controller instruction caching. 1 Disable flash controller instruction caching.
11 DFCDA	Disable Flash Controller Data Caching Disables flash controller data caching. 0 Enable flash controller data caching 1 Disable flash controller data caching.
10 CFCC	Clear Flash Controller Cache Writing a 1 to this field clears the cache. Writing a 0 to this field is ignored. This field always reads as 0.
9 ARB	Arbitration select 0 Fixed-priority arbitration for the crossbar masters 1 Round-robin arbitration for the crossbar masters
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

### 16.2.4 Compute Operation Control Register (MCM\_CPO)

This register controls the Compute Operation.

Address: F000\_3000h base + 40h offset = F000\_3040h



**MCM\_CPO field descriptions**

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CPOWOI	Compute Operation Wake-up on Interrupt 0 No effect. 1 When set, the CPOREQ is cleared on any interrupt or exception vector fetch.
1 CPOACK	Compute Operation Acknowledge 0 Compute operation entry has not completed or compute operation exit has completed. 1 Compute operation entry has completed or compute operation exit has not completed.
0 CPOREQ	Compute Operation Request This bit is auto-cleared by vector fetching if CPOWOI = 1.

Table continues on the next page...

**MCM\_CPO field descriptions (continued)**

<b>Field</b>	<b>Description</b>
0	Request is cleared.
1	Request Compute Operation.





# Chapter 17

## Crossbar Switch Lite (AXBS-Lite)

### 17.1 Chip-specific AXBS-Lite information

#### 17.1.1 Crossbar-light switch master assignments

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

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

#### 17.1.2 Crossbar switch slave assignments

This device contains 3 slaves connected to the crossbar switch.

The slave assignment is as follows:

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

### 17.2 Introduction

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.

### 17.2.1 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 (see the chip-specific information).

## 17.3 Memory Map / Register Definition

This crossbar switch is designed for minimal gate count. It, therefore, has no memory-mapped configuration registers.

Please see the chip-specific information for information on whether the arbitration method in the crossbar switch is programmable, and by which module.

## 17.4 Functional Description

### 17.4.1 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, other than the flash (if present), is being idled by the crossbar, it remains parked with the last master to use the slave port. This is done to save the initial clock of arbitration delay that otherwise would be seen if the same master had to arbitrate to gain control of the slave port.

If present, the flash slave port parks on the CPU master whenever there is an idle flash slave port cycle. This is done to save the CPU the initial clock of arbitration delay that would be seen if the CPU had to gain control of the flash slave port.

## 17.4.2 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round-robin

The selection of the global slave port arbitration is controlled by the MCM module. For fixed priority, set MCM\_PLACR[ARB] or MCM\_CPCR[CBRR] to 0. For round robin, set MCM\_PLACR[ARB] or MCM\_CPCR[CBRR] to 1. This arbitration setting applies to all slave ports.

### 17.4.2.1 Arbitration during undefined length bursts

All lengths of burst accesses lock out arbitration until the last beat of the burst.

### 17.4.2.2 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level with the highest numbered master having the highest priority (for example, in a system with 5 masters, master 1 has lower priority than master 3). If two masters request access to the same slave port, the master with the highest priority gains control over the slave port.

#### NOTE

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

**Table 17-1. 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
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>

### 17.4.2.3 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

The round-robin arbitration mode generally provides a more fair allocation of the available slave-port bandwidth (compared to fixed priority) as the fixed master priority does not affect the master selection.

## 17.5 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 18

## Low-Leakage Wakeup Unit (LLWU)

### 18.1 Chip-specific LLWU information

This device uses the 8 external wakeup pin inputs and 4 internal modules as wakeup sources to the LLWU module.

#### 18.1.1 LLWU interrupt

The interrupt requests of external sources and internal peripherals are used in wakeup.

#### NOTE

Do not mask the LLWU interrupt when in LLS/VLLSx mode. Masking the interrupt prevents the device from exiting stop mode when a wakeup is detected.

#### 18.1.2 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 18-1. Wakeup Source**

LLWU pin	Module source or pin name
LLWU_P5	PTB0
LLWU_P6	PTC1

*Table continues on the next page...*

**Table 18-1. Wakeup Source (continued)**

LLWU pin	Module source or pin name
LLWU_P7	PTC3
LLWU_P8	PTC4
LLWU_P9	PTC5
LLWU_P10	PTC6
LLWU_P14	PTD4
LLWU_P15	PTD6
LLWU_M0IF	LPTMR0
LLWU_M1IF	CMP0
LLWU_M2IF	Reserved
LLWU_M3IF	Reserved
LLWU_M4IF	Reserved
LLWU_M5IF	RTC Alarm
LLWU_M6IF	Reserved
LLWU_M7IF	RTC Seconds

## 18.2 Introduction

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 MCUs](#) for further details on using the LLWU.

### 18.2.1 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.

## 18.2.2 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].

### 18.2.2.1 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.

### 18.2.2.2 VLLS modes

All wakeup and reset events result in VLLS exit via a reset flow.

### 18.2.2.3 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.

### 18.2.2.4 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.

### 18.2.3 Block diagram

The following figure is the block diagram for the LLWU module.

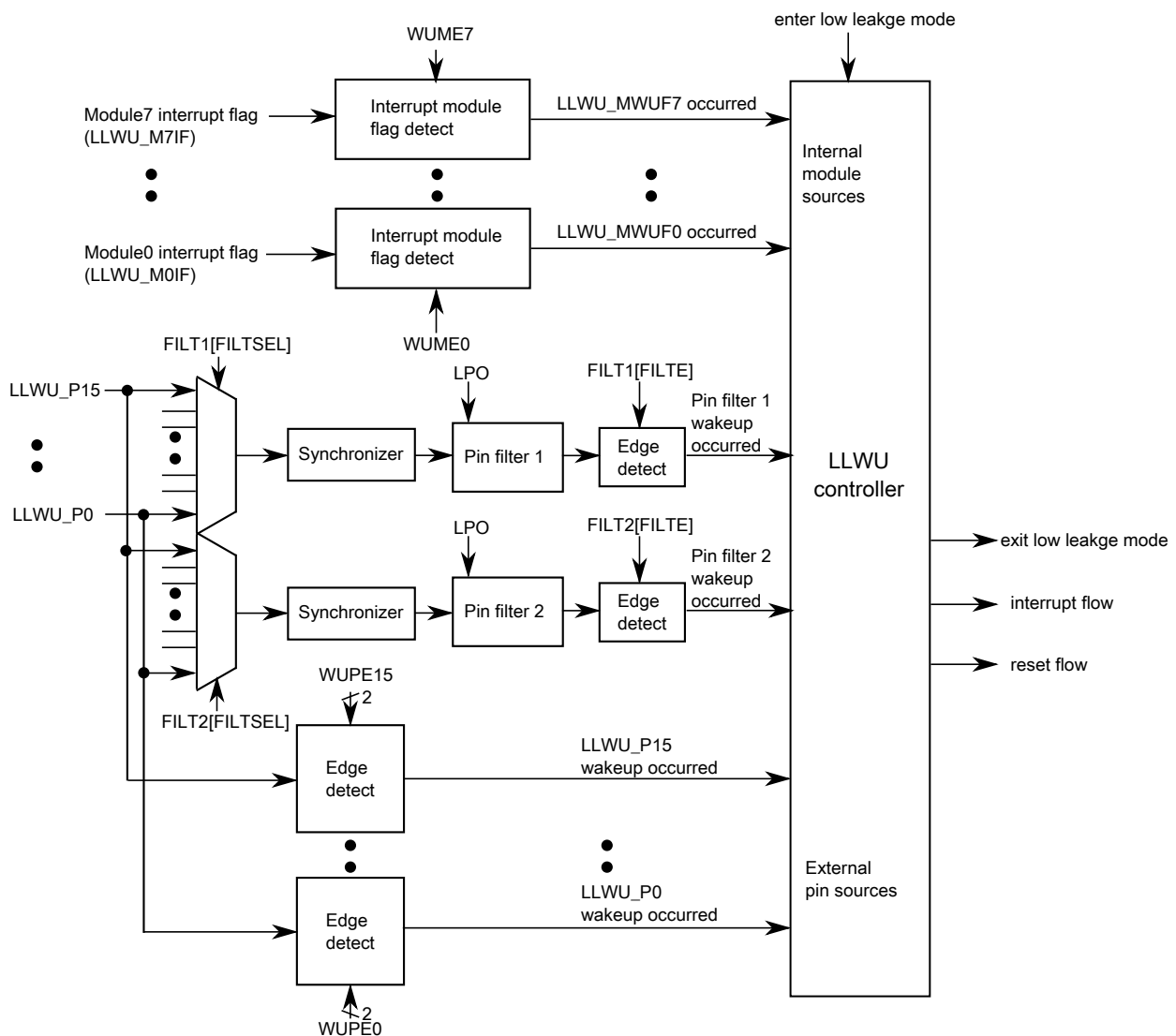


Figure 18-1. LLWU block diagram

## 18.3 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 18-2. LLWU signal descriptions**

Signal	Description	I/O
LLWU_Pn	Wakeup inputs (n = 0-15 )	I

## 18.4 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="#">18.4.1/260</a>
4007_C001	LLWU Pin Enable 2 register (LLWU_PE2)	8	R/W	00h	<a href="#">18.4.2/261</a>
4007_C002	LLWU Pin Enable 3 register (LLWU_PE3)	8	R/W	00h	<a href="#">18.4.3/262</a>
4007_C003	LLWU Pin Enable 4 register (LLWU_PE4)	8	R/W	00h	<a href="#">18.4.4/263</a>
4007_C004	LLWU Module Enable register (LLWU_ME)	8	R/W	00h	<a href="#">18.4.5/264</a>
4007_C005	LLWU Flag 1 register (LLWU_F1)	8	R/W	00h	<a href="#">18.4.6/266</a>
4007_C006	LLWU Flag 2 register (LLWU_F2)	8	R/W	00h	<a href="#">18.4.7/268</a>
4007_C007	LLWU Flag 3 register (LLWU_F3)	8	R	00h	<a href="#">18.4.8/269</a>
4007_C008	LLWU Pin Filter 1 register (LLWU_FILT1)	8	R/W	00h	<a href="#">18.4.9/271</a>
4007_C009	LLWU Pin Filter 2 register (LLWU_FILT2)	8	R/W	00h	<a href="#">18.4.10/272</a>

### 18.4.1 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
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

**18.4.2 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
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

**18.4.3 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**

<b>Field</b>	<b>Description</b>
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>
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>

**18.4.4 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.

## Memory map/register definition

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>
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>

## 18.4.5 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



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>

## 18.4.6 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

## 18.4.7 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>

**18.4.8 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

## 18.4.9 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

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\_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.
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

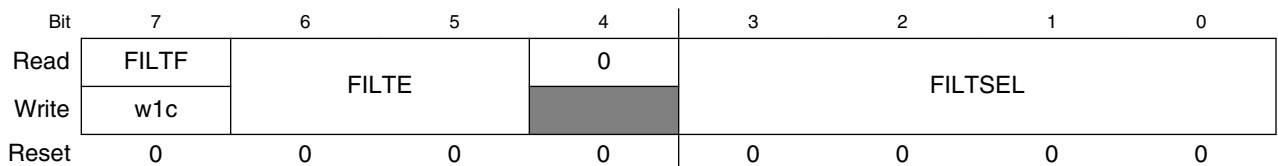
**18.4.10 LLWU Pin Filter 2 register (LLWU\_FILT2)**

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



**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.
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

## 18.5 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.

### 18.5.1 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.

### 18.5.2 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.

### 18.5.3 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 19

## Peripheral Bridge (AIPS-Lite)

### 19.1 Chip-specific AIPS-Lite information

#### 19.1.1 Number of peripheral bridges

This device contains one peripheral bridge.

#### 19.1.2 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.

### 19.2 Introduction

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.

#### 19.2.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width

## 19.2.2 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
- 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.

The AIPS-Lite module uses the data width of accessed peripheral to perform proper data byte lane routing; bus decomposition (bus sizing) is performed when the access size is larger than the peripheral's data width.

## 19.3 Memory map/register definition

AIPS memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	Master Privilege Register A (AIPS_MPRA)	32	R/W	<a href="#">See section</a>	<a href="#">19.3.1/277</a>
20	Peripheral Access Control Register (AIPS_PACRA)	32	R/W	<a href="#">See section</a>	<a href="#">19.3.2/279</a>
24	Peripheral Access Control Register (AIPS_PACRB)	32	R/W	<a href="#">See section</a>	<a href="#">19.3.2/279</a>
28	Peripheral Access Control Register (AIPS_PACRC)	32	R/W	<a href="#">See section</a>	<a href="#">19.3.2/279</a>
2C	Peripheral Access Control Register (AIPS_PACRD)	32	R/W	<a href="#">See section</a>	<a href="#">19.3.2/279</a>
40	Peripheral Access Control Register (AIPS_)	32	R/W	<a href="#">See section</a>	<a href="#">19.3.3/ 0</a>

### 19.3.1 Master Privilege Register A (AIPS\_MPRA)

The MPRA specifies identical 4-bit fields defining the access-privilege level associated with a bus master to various peripherals on the chip. The register provides one field per bus master.

#### NOTE

At reset, the default value loaded into the MPRA fields is chip-specific. See the chip configuration details for the value of a particular device.

A register field that maps to an unimplemented master or peripheral behaves as read-only-zero.

Each master is assigned a logical ID from 0 to 15. See the master logical ID assignment table in the chip-specific AIPS information.

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	MTR0	MTW0	MPL0	0	MTR2	MTW2	MPL2	0	MTR3	MTW3	MPL3				
W																
Reset	0*	1*	1*	1*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

\* Notes:

- The reset value is chip-dependent and can be found in the chip-specific AIPS information.

#### AIPS\_MPRA field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 MTR0	Master 0 Trusted For Read  Determines whether the master is trusted for read accesses.  0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.

Table continues on the next page...

## AIPS\_MPRA field descriptions (continued)

Field	Description
29 MTW0	<p>Master 0 Trusted For Writes</p> <p>Determines whether the master is trusted for write accesses.</p> <p>0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.</p>
28 MPL0	<p>Master 0 Privilege Level</p> <p>Specifies how the privilege level of the master is determined.</p> <p>0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.</p>
23 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
22 MTR2	<p>Master 2 Trusted For Read</p> <p>Determines whether the master is trusted for read accesses.</p> <p>0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.</p>
21 MTW2	<p>Master 2 Trusted For Writes</p> <p>Determines whether the master is trusted for write accesses.</p> <p>0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.</p>
20 MPL2	<p>Master 2 Privilege Level</p> <p>Specifies how the privilege level of the master is determined.</p> <p>0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.</p>
19 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
18 MTR3	<p>Master 3 Trusted For Read</p> <p>Determines whether the master is trusted for read accesses.</p> <p>0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.</p>
17 MTW3	<p>Master 3 Trusted For Writes</p> <p>Determines whether the master is trusted for write accesses.</p> <p>0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.</p>
16 MPL3	<p>Master 3 Privilege Level</p> <p>Specifies how the privilege level of the master is determined.</p> <p>0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.</p>

## 19.3.2 Peripheral Access Control Register (AIPS\_PACR<sub>n</sub>)

The following table shows the location of each peripheral slot's PACR field in the PACR registers.

Offset	Register	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
0x20	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x24	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x28	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x2C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31

Address: 0h base + 20h offset + (4d × 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	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
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	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

\* Notes:

- The reset value is chip-dependent and can be found in the AIPS chip-specific information.

### AIPS\_PACR<sub>n</sub> field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 SP0	Supervisor Protect  Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL <sub>n</sub> ] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates.  0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
29 WP0	Write Protect  Determines whether the peripheral allows write accesses. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates.  0 This peripheral allows write accesses. 1 This peripheral is write protected.
28 TP0	Trusted Protect

Table continues on the next page...

AIPS\_PACR<sub>n</sub> field descriptions (continued)

Field	Description
	<p>Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.</p> <p>0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.</p>
27 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
26 SP1	<p>Supervisor Protect</p> <p>Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR<sub>x</sub>[MPL<sub>n</sub>] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.</p>
25 WP1	<p>Write Protect</p> <p>Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral allows write accesses. 1 This peripheral is write protected.</p>
24 TP1	<p>Trusted Protect</p> <p>Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.</p> <p>0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.</p>
23 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
22 SP2	<p>Supervisor Protect</p> <p>Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR<sub>x</sub>[MPL<sub>n</sub>] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.</p>
21 WP2	<p>Write Protect</p> <p>Determines whether the peripheral allows write accesses. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral allows write accesses. 1 This peripheral is write protected.</p>
20 TP2	<p>Trusted Protect</p>

*Table continues on the next page...*



AIPS\_PACR<sub>n</sub> field descriptions (continued)

Field	Description
	<p>Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.</p> <p>0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.</p>
19 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
18 SP3	<p>Supervisor Protect</p> <p>Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL<sub>n</sub>] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.</p>
17 WP3	<p>Write Protect</p> <p>Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral allows write accesses. 1 This peripheral is write protected.</p>
16 TP3	<p>Trusted Protect</p> <p>Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.</p> <p>0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.</p>
15 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
14 SP4	<p>Supervisor Protect</p> <p>Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL<sub>n</sub>] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.</p>
13 WP4	<p>Write Protect</p> <p>Determines whether the peripheral allows write accesses. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral allows write accesses. 1 This peripheral is write protected.</p>
12 TP4	<p>Trusted Protect</p>

*Table continues on the next page...*

**AIPS\_PACR<sub>n</sub> field descriptions (continued)**

Field	Description
	<p>Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.</p> <p>0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.</p>
11 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
10 SP5	<p>Supervisor Protect</p> <p>Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR<sub>x</sub>[MPL<sub>n</sub>] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.</p>
9 WP5	<p>Write Protect</p> <p>Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral allows write accesses. 1 This peripheral is write protected.</p>
8 TP5	<p>Trusted Protect</p> <p>Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.</p> <p>0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.</p>
7 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
6 SP6	<p>Supervisor Protect</p> <p>Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR<sub>x</sub>[MPL<sub>n</sub>] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.</p>
5 WP6	<p>Write Protect</p> <p>Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral allows write accesses. 1 This peripheral is write protected.</p>
4 TP6	<p>Trusted Protect</p>

*Table continues on the next page...*

**AIPS\_PACR<sub>n</sub> field descriptions (continued)**

Field	Description
	<p>Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.</p> <p>0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.</p>
3 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
2 SP7	<p>Supervisor Protect</p> <p>Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPRx[MPL<sub>n</sub>] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.</p>
1 WP7	<p>Write Protect</p> <p>Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates.</p> <p>0 This peripheral allows write accesses. 1 This peripheral is write protected.</p>
0 TP7	<p>Trusted Protect</p> <p>Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.</p> <p>0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.</p>

## 19.4 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.

## 19.4.1 Access support

All combinations of access size and peripheral data port width are supported. An access that is larger than the target peripheral's data width will be decomposed to multiple, smaller accesses. Bus decomposition is terminated by a transfer error caused by an access to an empty register area.

# Chapter 20

## Direct Memory Access Multiplexer (DMAMUX)

### 20.1 Chip-specific DMAMUX information

#### 20.1.1 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 no hard correlation between any of the DMA request sources and a specific DMA channel. Some of the modules support Asynchronous DMA operation as indicated by the last column in the following DMA source assignment table.

**Table 20-1. 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	LPUART0	Receive	Yes
3	LPUART0	Transmit	Yes
4	LPUART1	Receive	Yes
5	LPUART1	Transmit	Yes
6	UART2	Receive	
7	UART2	Transmit	
8	Reserved	—	
9	Reserved	—	
10	Flex_IO	FlexIO_Ch0	Yes
11	Flex_IO	FlexIO_Ch1	Yes
12	Flex_IO	FlexIO_Ch2	Yes
13	Flex_IO	FlexIO_Ch3	Yes
14	Reserved	—	

*Table continues on the next page...*

**Table 20-1. DMA request sources - MUX 0 (continued)**

Source number	Source module	Source description	Async DMA capable
15	Reserved	—	
16	SPI0	Receive	
17	SPI0	Transmit	
18	SPI1	Receive	
19	SPI1	Transmit	
20	Reserved	—	
21	Reserved	—	
22	I <sup>2</sup> C0	I2C transfer done	
23	I <sup>2</sup> C1	I2C transfer done	
24	TPM0	Channel 0	Yes
25	TPM0	Channel 1	Yes
26	TPM0	Channel 2	Yes
27	TPM0	Channel 3	Yes
28	TPM0	Channel 4	Yes
29	TPM0	Channel 5	Yes
30	Reserved	—	
31	Reserved	—	
32	TPM1	Channel 0	Yes
33	TPM1	Channel 1	Yes
34	TPM2	Channel 0	Yes
35	TPM2	Channel 1	Yes
36	Reserved	—	
37	Reserved	—	
38	Reserved	—	
39	Reserved	—	
40	ADC0	ADC conversion complete	Yes
41	Reserved	—	
42	CMP0	Rising or falling edge detected on COUT	Yes
43	Reserved	—	
44	Reserved	—	
45	DAC0	—	
46	Reserved	—	
47	Reserved	—	
48	Reserved	—	
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

*Table continues on the next page...*

**Table 20-1. DMA request sources - MUX 0 (continued)**

Source number	Source module	Source description	Async DMA capable
54	TPM0	Overflow	Yes
55	TPM1	Overflow	Yes
56	TPM2	Overflow	Yes
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.

## 20.1.2 DMA transfers via PIT trigger

The PIT module can trigger a DMA transfer on the first two DMA channels. The assignments are detailed at [PIT/DMA periodic trigger assignments](#) .

## 20.2 Introduction

### 20.2.1 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.

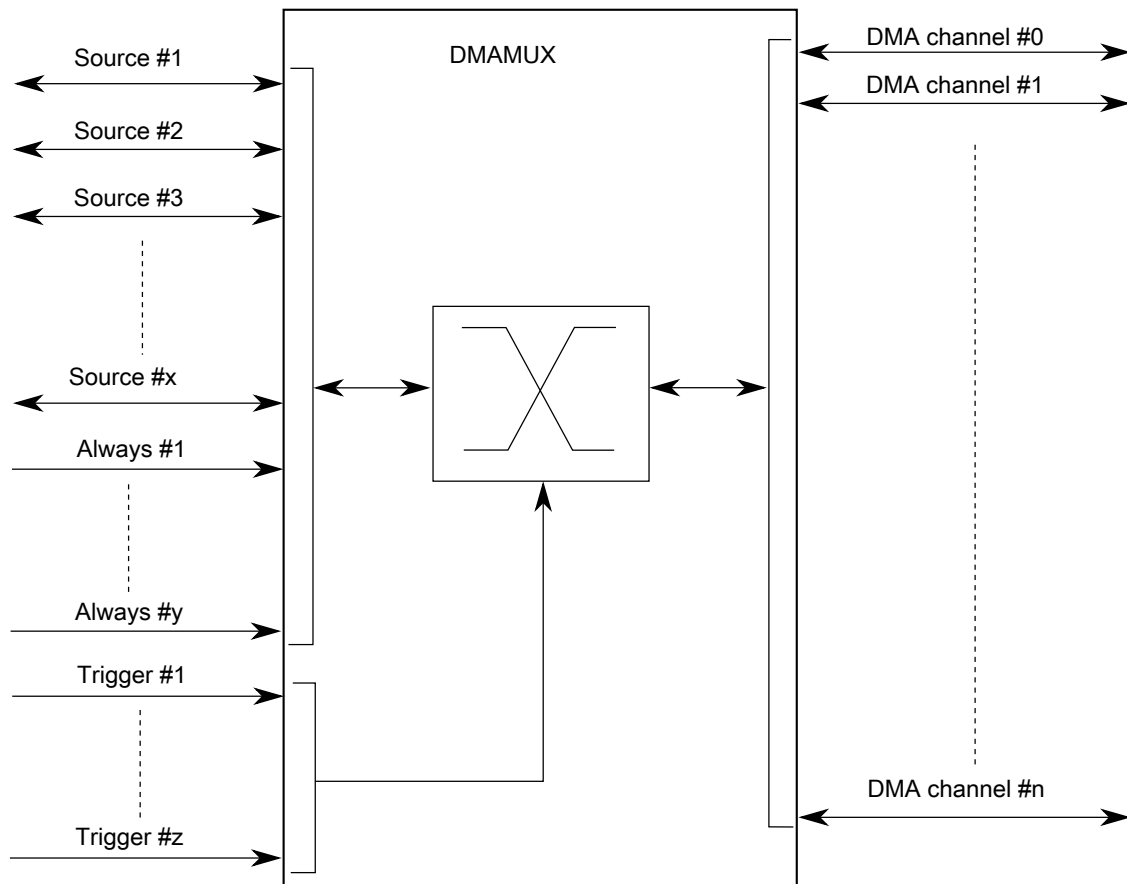


Figure 20-1. DMAMUX block diagram

## 20.2.2 Features

The DMAMUX module provides these features:

- Up to 63 peripheral slots and up to four always-on slots can be routed to four channels.
- four independently selectable DMA channel routers.
  - The first two 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.

## 20.2.3 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 0–1.

## 20.3 External signal description

The DMAMUX has no external pins.

## 20.4 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 (DMAMUX0_CHCFG0)	8	R/W	00h	<a href="#">20.4.1/289</a>
4002_1001	Channel Configuration register (DMAMUX0_CHCFG1)	8	R/W	00h	<a href="#">20.4.1/289</a>
4002_1002	Channel Configuration register (DMAMUX0_CHCFG2)	8	R/W	00h	<a href="#">20.4.1/289</a>
4002_1003	Channel Configuration register (DMAMUX0_CHCFG3)	8	R/W	00h	<a href="#">20.4.1/289</a>

### 20.4.1 Channel Configuration register (DMAMUXx\_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

**DMAMUXx\_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>
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>

**20.5 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:

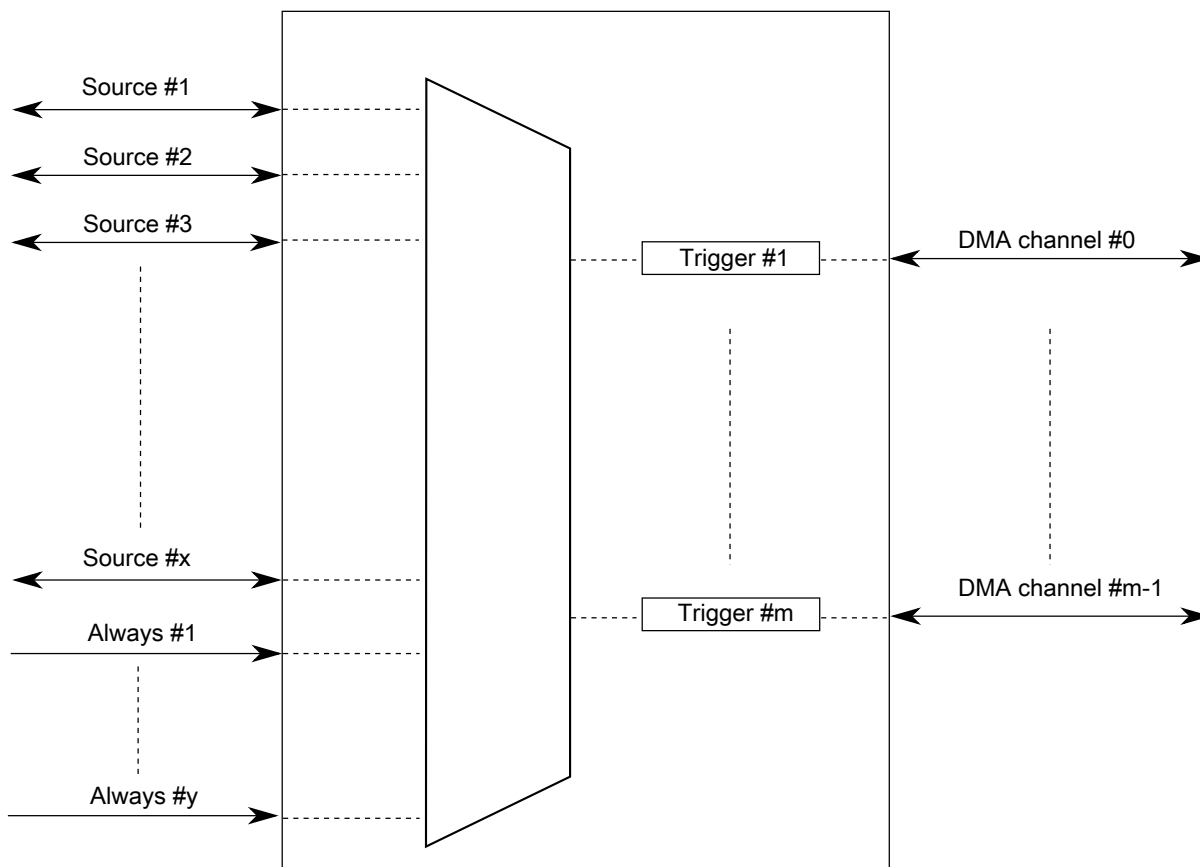
- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

### 20.5.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 2 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.

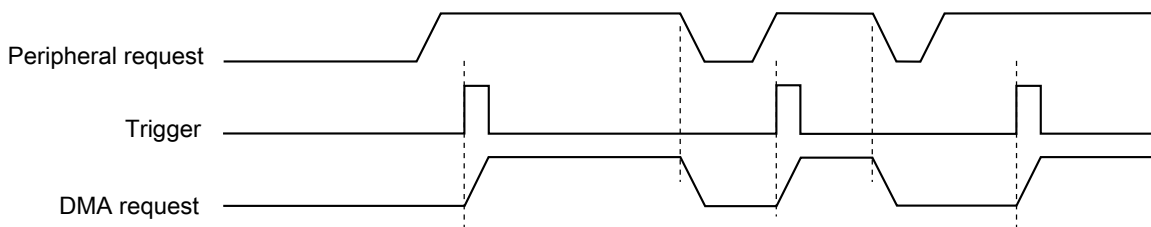
#### Note

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.



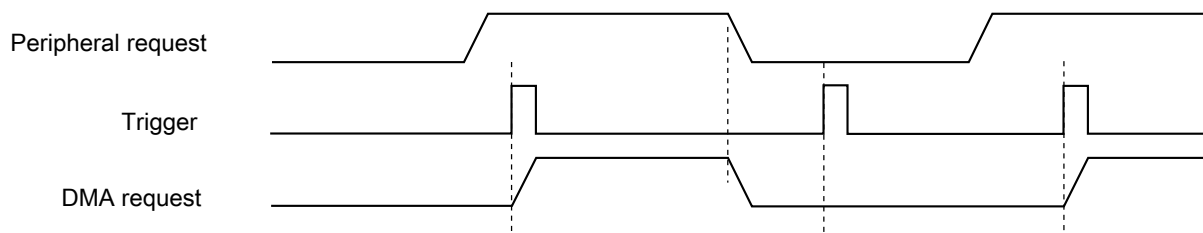
**Figure 20-2. DMAMUX triggered channels**

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 20-3. 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 20-4. 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  $\mu\text{s}$  (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.

## 20.5.2 DMA channels with no triggering capability

The other channels of the DMAMUX provide the normal routing functionality as described in [Modes of operation](#).

### 20.5.3 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.

## 20.6 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

### 20.6.1 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.

### 20.6.2 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 2 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 2 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 21

## DMA Controller Module

### 21.1 Introduction

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

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

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

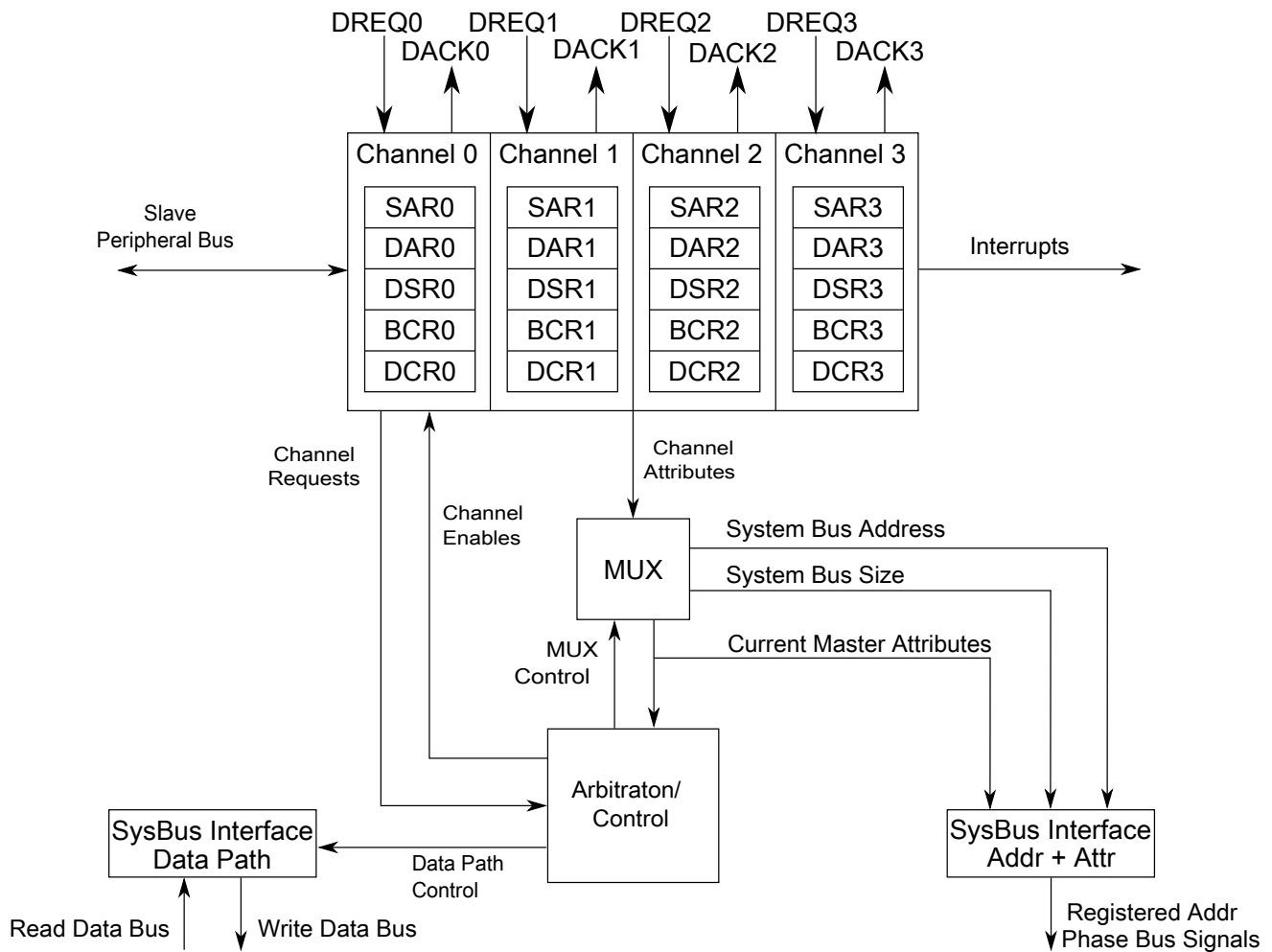
#### Note

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

#### 21.1.1 Overview

The DMA controller module enables fast transfers of data, providing an efficient way to move blocks of data with minimal processor interaction. The DMA module, shown in the following figure, has four channels that allow 8-bit, 16-bit, or 32-bit data transfers. Each channel has a dedicated Source Address register ( $SAR_n$ ), Destination Address register ( $DAR_n$ ), Status register ( $DSR_n$ ), Byte Count register ( $BCR_n$ ), and Control register ( $DCR_n$ ). Collectively, the combined program-visible registers associated with each channel define a transfer control descriptor (TCD). All transfers are dual address, moving data from a source memory location to a destination memory location with the module operating as a 32-bit bus master connected to the system bus. The programming model is accessed through a 32-bit connection with the slave peripheral bus. DMA data transfers may be explicitly initiated by software or by peripheral hardware requests.

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



**Figure 21-1. 4-Channel DMA Block Diagram**

The terms *peripheral request* and *DREQ* refer to a DMA request from one of the on-chip peripherals or package pins. The DMA provides hardware handshake signals: either a DMA acknowledge (DACK) or a done indicator back to the peripheral.

### 21.1.2 Features

The DMA controller module features:

- Four independently programmable DMA controller channels
- Dual-address transfers via 32-bit master connection to the system bus
- Data transfers in 8-, 16-, or 32-bit blocks
- Continuous-mode or cycle-steal transfers from software or peripheral initiation

- Automatic hardware acknowledge/done indicator from each channel
- Independent source and destination address registers
- Optional modulo addressing and automatic updates of source and destination addresses
- Independent transfer sizes for source and destination
- Optional auto-alignment feature for source or destination accesses
- Optional automatic single or double channel linking
- Programming model accessed via 32-bit slave peripheral bus
- Channel arbitration on transfer boundaries using fixed priority scheme

## 21.2 DMA Transfer Overview

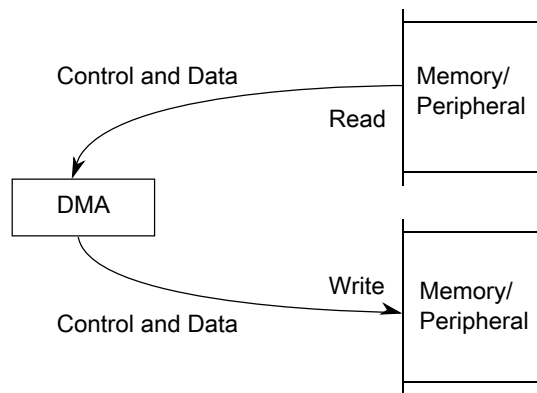
The DMA module can move data within system memory (including memory and peripheral devices) with minimal processor intervention, greatly improving overall system performance.

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

As soon as a channel has been initialized, it may be started by setting  $DCR_n[START]$  or a properly-selected peripheral DMA request, depending on the status of  $DCR_n[ERQ]$ .

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

- Dual-address transfers—A dual-address transfer consists of a read followed by a write and is initiated by a request using the  $DCR_n[START]$  bit or by a peripheral DMA request. The read data is temporarily held in the DMA channel hardware until the write operation. Two types of single transfers occur: a read from a source address followed by a write to a destination address. See the following figure.



**Figure 21-2. Dual-Address Transfer**

Any operation involving a DMA channel follows the same three steps:

1. Channel initialization—The transfer control descriptor, contained in the channel registers, is loaded with address pointers, a byte-transfer count, and control information using accesses from the slave peripheral bus.
2. Data transfer—The DMA accepts requests for data transfers. Upon receipt of a request, it provides address and bus control for the transfers via its master connection to the system bus and temporary storage for the read data. The channel performs one or more source read and destination write data transfers.
3. Channel termination—Occurs after the operation is finished successfully or due to an error. The channel indicates the operation status in the channel's DSR, described in the definitions of the DMA Status Registers (DSR<sub>n</sub>) and Byte Count Registers (BCR<sub>n</sub>).

## 21.3 Memory Map/Register Definition

Information about the registers related to the DMA controller module can be found here.

Descriptions of each register and its bit assignments follow. Modifying DMA control registers during a transfer can result in undefined operation. The following table shows the mapping of DMA controller registers. The DMA programming model is accessed via the slave peripheral bus. The concatenation of the source and destination address registers, the status and byte count register, and the control register create a 128-bit transfer control descriptor (TCD) that defines the operation of each DMA channel.

## DMA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4000_8100	Source Address Register (DMA_SAR0)	32	R/W	0000_0000h	<a href="#">21.3.1/303</a>
4000_8104	Destination Address Register (DMA_DAR0)	32	R/W	0000_0000h	<a href="#">21.3.2/304</a>
4000_8108	DMA Status Register / Byte Count Register (DMA_DSR_BCR0)	32	R/W	0000_0000h	<a href="#">21.3.3/305</a>
4000_810C	DMA Control Register (DMA_DCR0)	32	R/W	0000_0000h	<a href="#">21.3.4/307</a>
4000_8110	Source Address Register (DMA_SAR1)	32	R/W	0000_0000h	<a href="#">21.3.1/303</a>
4000_8114	Destination Address Register (DMA_DAR1)	32	R/W	0000_0000h	<a href="#">21.3.2/304</a>
4000_8118	DMA Status Register / Byte Count Register (DMA_DSR_BCR1)	32	R/W	0000_0000h	<a href="#">21.3.3/305</a>
4000_811C	DMA Control Register (DMA_DCR1)	32	R/W	0000_0000h	<a href="#">21.3.4/307</a>
4000_8120	Source Address Register (DMA_SAR2)	32	R/W	0000_0000h	<a href="#">21.3.1/303</a>
4000_8124	Destination Address Register (DMA_DAR2)	32	R/W	0000_0000h	<a href="#">21.3.2/304</a>
4000_8128	DMA Status Register / Byte Count Register (DMA_DSR_BCR2)	32	R/W	0000_0000h	<a href="#">21.3.3/305</a>
4000_812C	DMA Control Register (DMA_DCR2)	32	R/W	0000_0000h	<a href="#">21.3.4/307</a>
4000_8130	Source Address Register (DMA_SAR3)	32	R/W	0000_0000h	<a href="#">21.3.1/303</a>
4000_8134	Destination Address Register (DMA_DAR3)	32	R/W	0000_0000h	<a href="#">21.3.2/304</a>
4000_8138	DMA Status Register / Byte Count Register (DMA_DSR_BCR3)	32	R/W	0000_0000h	<a href="#">21.3.3/305</a>
4000_813C	DMA Control Register (DMA_DCR3)	32	R/W	0000_0000h	<a href="#">21.3.4/307</a>

### 21.3.1 Source Address Register (DMA\_SAR<sub>n</sub>)

#### Restriction

For this register:

- Only 32-bit writes are allowed. 16-bit and 8-bit writes result in a bus error.
- Only several values are allowed to be written to bits 31-20 of this register, see the value list in the field description. A write of any other value to these bits causes a configuration error when the channel starts to execute. For more information about the configuration error, see the description of the **CE** field of DSR.

Address: 4000\_8000h base + 100h 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	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

### DMA\_SARn field descriptions

Field	Description
SAR	<p>SAR</p> <p>Each SAR contains the byte address used by the DMA controller to read data. The SARn is typically aligned on a 0-modulo-ssize boundary—that is, on the natural alignment of the source data.</p> <p><b>Restriction:</b> Bits 31-20 of this register must be written with one of only several allowed values. Each of these allowed values corresponds to a valid region of the device's memory map. The allowed values are:</p> <ul style="list-style-type: none"> <li>• 0x000x_xxxx</li> <li>• 0x1FFx_xxxx</li> <li>• 0x200x_xxxx</li> <li>• 0x400x_xxxx</li> </ul> <p>After being written with one of the allowed values, bits 31-20 read back as the written value. After being written with any other value, bits 31-20 read back as an indeterminate value.</p>

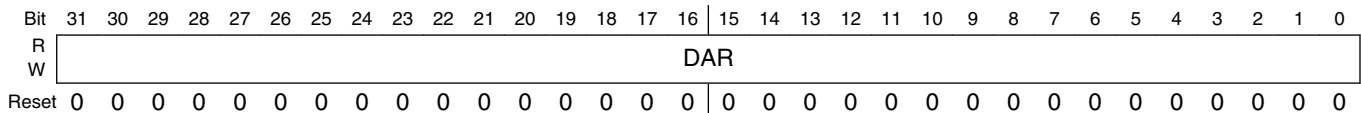
### 21.3.2 Destination Address Register (DMA\_DARn)

#### Restriction

For this register:

- Only 32-bit writes are allowed. 16-bit and 8-bit writes result in a bus error.
- Only several values are allowed to be written to bits 31-20 of this register, see the value list in the field description. A write of any other value to these bits causes a configuration error when the channel starts to execute. For more information about the configuration error, see the description of the [CE](#) field of DSR.

Address: 4000\_8000h base + 104h offset + (16d × i), where i=0d to 3d



### DMA\_DARn field descriptions

Field	Description
DAR	<p>DAR</p> <p>Each DAR contains the byte address used by the DMA controller to write data. The DARn is typically aligned on a 0-modulo-dsize boundary—that is, on the natural alignment of the destination data.</p> <p><b>Restriction:</b> Bits 31-20 of this register must be written with one of only several allowed values. Each of these allowed values corresponds to a valid region of the device's memory map. The allowed values are:</p>



DMA\_DAR<sub>n</sub> field descriptions (continued)

Field	Description
	<ul style="list-style-type: none"> <li>• 0x000x_xxxx</li> <li>• 0x1FFx_xxxx</li> <li>• 0x200x_xxxx</li> <li>• 0x400x_xxxx</li> </ul> <p>After being written with one of the allowed values, bits 31-20 read back as the written value. After being written with any other value, bits 31-20 read back as an indeterminate value.</p>

21.3.3 DMA Status Register / Byte Count Register (DMA\_DSR\_BCR<sub>n</sub>)

DSR and BCR are two logical registers that occupy one 32-bit address. DSR<sub>n</sub> occupies bits 31–24, and BCR<sub>n</sub> occupies bits 23–0. DSR<sub>n</sub> contains flags indicating the channel status, and BCR<sub>n</sub> contains the number of bytes yet to be transferred for a given block.

On the successful completion of the write transfer, BCR<sub>n</sub> decrements by 1, 2, or 4 for 8-bit, 16-bit, or 32-bit accesses, respectively. BCR<sub>n</sub> is cleared if a 1 is written to DSR[**DONE**].

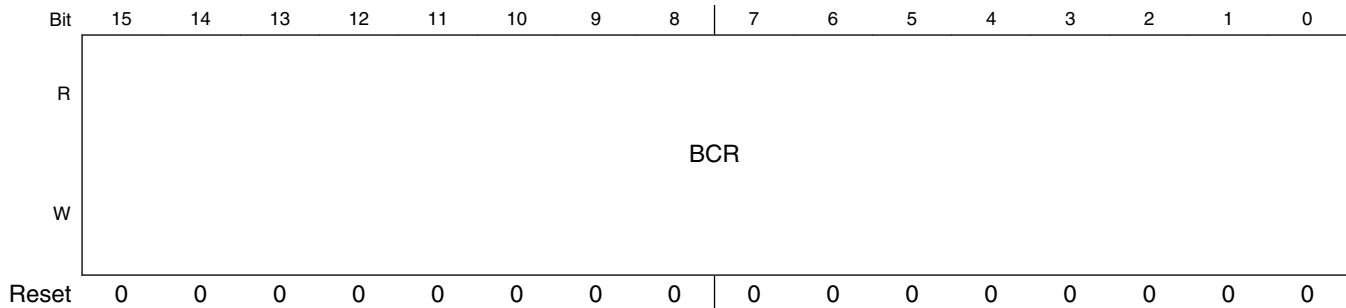
In response to an event, the DMA controller writes to the appropriate DSR<sub>n</sub> bit. Only a write to DSR<sub>n</sub>[**DONE**] results in action. DSR<sub>n</sub>[**DONE**] is set when the block transfer is complete.

When a transfer sequence is initiated and BCR<sub>n</sub>[**BCR**] is not a multiple of 4 or 2 when the DMA is configured for 32-bit or 16-bit transfers, respectively, DSR<sub>n</sub>[**CE**] is set and no transfer occurs.

Address: 4000\_8000h 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	CE	BES	BED	0	REQ	BSY	DONE	BCR								
W																	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

## Memory Map/Register Definition



### DMA\_DSR\_BCRn field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CE	<p>Configuration Error</p> <p>Any of the following conditions causes a configuration error:</p> <ul style="list-style-type: none"> <li>• BCR, SAR, or DAR does not match the requested transfer size.</li> <li>• A value greater than 0F_FFFFh is written to BCR.</li> <li>• Bits 31-20 of SAR or DAR are written with a value other than one of the allowed values. See <a href="#">SAR</a> and <a href="#">DAR</a>.</li> <li>• SSIZE or DSIZE is set to an unsupported value.</li> <li>• BCR equals 0 when the DMA receives a start condition.</li> </ul> <p>CE is cleared at hardware reset or by writing a 1 to DONE.</p> <p>0 No configuration error exists. 1 A configuration error has occurred.</p>
29 BES	<p>Bus Error on Source</p> <p>BES is cleared at hardware reset or by writing a 1 to DONE.</p> <p>0 No bus error occurred. 1 The DMA channel terminated with a bus error during the read portion of a transfer.</p>
28 BED	<p>Bus Error on Destination</p> <p>BED is cleared at hardware reset or by writing a 1 to DONE.</p> <p>0 No bus error occurred. 1 The DMA channel terminated with a bus error during the write portion of a transfer.</p>
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 REQ	<p>Request</p> <p>0 No request is pending or the channel is currently active. Cleared when the channel is selected. 1 The DMA channel has a transfer remaining and the channel is not selected.</p>
25 BSY	<p>Busy</p> <p>0 DMA channel is inactive. Cleared when the DMA has finished the last transaction. 1 BSY is set the first time the channel is enabled after a transfer is initiated.</p>
24 DONE	Transactions Done

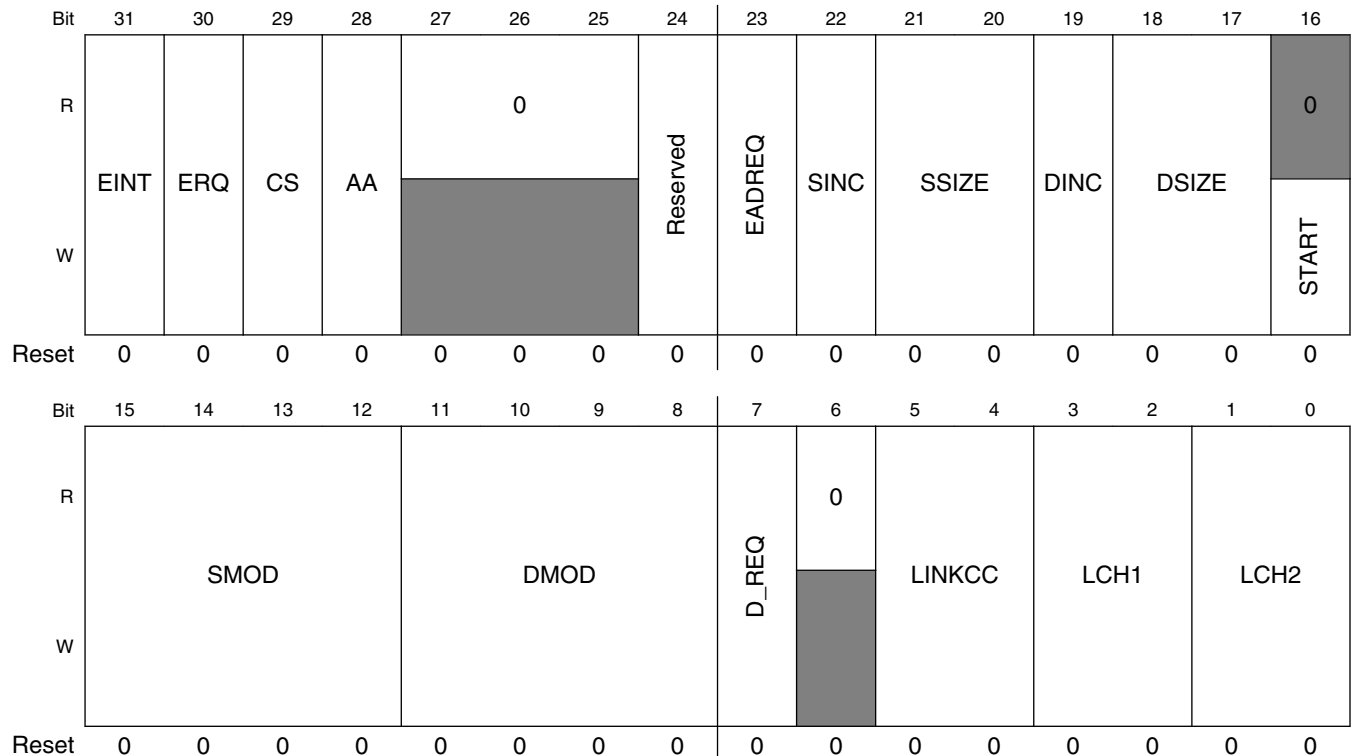
*Table continues on the next page...*

**DMA\_DSR\_BCRn field descriptions (continued)**

Field	Description
	<p>Set when all DMA controller transactions complete as determined by transfer count, or based on error conditions. When BCR reaches 0, DONE is set when the final transfer completes successfully. DONE can also be used to abort a transfer by resetting the status bits. When a transfer completes, software must clear DONE before reprogramming the DMA.</p> <p>0 DMA transfer is not yet complete. Writing a 0 has no effect.                      1 DMA transfer completed. Writing a 1 to this bit clears all DMA status bits and should be used in an interrupt service routine to clear the DMA interrupt and error bits.</p>
BCR	<p>BCR</p> <p>This field contains the number of bytes yet to be transferred for a given block.</p> <p><b>Restriction:</b> BCR must be written with a value equal to or less than 0F_FFFFh. After being written with a value in this range, bits 23-20 of BCR read back as 0000b. A write to BCR of a value greater than 0F_FFFFh causes a configuration error when the channel starts to execute. After being written with a value in this range, bits 23-20 of BCR read back as 0001b.</p>

**21.3.4 DMA Control Register (DMA\_DCRn)**

Address: 4000\_8000h base + 10Ch offset + (16d × i), where i=0d to 3d



**DMA\_DCRn field descriptions**

Field	Description
31 EINT	Enable Interrupt on Completion of Transfer

Table continues on the next page...

## DMA\_DCRn field descriptions (continued)

Field	Description
	<p>Determines whether an interrupt is generated by completing a transfer or by the occurrence of an error condition.</p> <p>0 No interrupt is generated. 1 Interrupt signal is enabled.</p>
30 ERQ	<p>Enable Peripheral Request</p> <p><b>CAUTION:</b> Be careful: a collision can occur between START and D_REQ when ERQ is 1.</p> <p>0 Peripheral request is ignored. 1 Enables peripheral request to initiate transfer. A software-initiated request (setting START) is always enabled.</p>
29 CS	<p>Cycle Steal</p> <p>0 DMA continuously makes read/write transfers until the BCR decrements to 0. 1 Forces a single read/write transfer per request.</p>
28 AA	<p>Auto-align</p> <p>AA and SIZE bits determine whether the source or destination is auto-aligned; that is, transfers are optimized based on the address and size.</p> <p>0 Auto-align disabled 1 If SSIZE indicates a transfer no smaller than DSIZE, source accesses are auto-aligned; otherwise, destination accesses are auto-aligned. Source alignment takes precedence over destination alignment. If auto-alignment is enabled, the appropriate address register increments, regardless of DINC or SINC.</p>
27–25 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
24 Reserved	<p>This field is reserved.</p> <p><b>CAUTION:</b> Must be written as zero; otherwise, undefined behavior results.</p>
23 EADREQ	<p>Enable asynchronous DMA requests</p> <p>Enables the channel to support asynchronous DREQs while the MCU is in Stop mode.</p> <p>0 Disabled 1 Enabled</p>
22 SINC	<p>Source Increment</p> <p>Controls whether the source address increments after each successful transfer.</p> <p>0 No change to SAR after a successful transfer. 1 The SAR increments by 1, 2, 4 as determined by the transfer size.</p>
21–20 SSIZE	<p>Source Size</p> <p>Determines the data size of the source bus cycle for the DMA controller.</p> <p>00 32-bit 01 8-bit</p>

Table continues on the next page...

## DMA\_DCRn field descriptions (continued)

Field	Description
	10 16-bit 11 Reserved (generates a configuration error (DSRn[CE]) if incorrectly specified at time of channel activation)
19 DINC	Destination Increment Controls whether the destination address increments after each successful transfer. 0 No change to the DAR after a successful transfer. 1 The DAR increments by 1, 2, 4 depending upon the size of the transfer.
18–17 DSIZE	Destination Size Determines the data size of the destination bus cycle for the DMA controller. 00 32-bit 01 8-bit 10 16-bit 11 Reserved (generates a configuration error (DSRn[CE]) if incorrectly specified at time of channel activation)
16 START	Start Transfer 0 DMA inactive 1 The DMA begins the transfer in accordance to the values in the TCDn. START is cleared automatically after one module clock and always reads as logic 0.
15–12 SMOD	Source Address Modulo Defines the size of the source data circular buffer used by the DMA Controller. If enabled (SMOD is non-zero), the buffer base address is located on a boundary of the buffer size. The value of this boundary is based upon the initial source address (SAR). The base address should be aligned to a 0-modulo-(circular buffer size) boundary. Misaligned buffers are not possible. The boundary is forced to the value determined by the upper address bits in the field selection.  0000 Buffer disabled 0001 Circular buffer size is 16 bytes. 0010 Circular buffer size is 32 bytes. 0011 Circular buffer size is 64 bytes. 0100 Circular buffer size is 128 bytes. 0101 Circular buffer size is 256 bytes. 0110 Circular buffer size is 512 bytes. 0111 Circular buffer size is 1 KB. 1000 Circular buffer size is 2 KB. 1001 Circular buffer size is 4 KB. 1010 Circular buffer size is 8 KB. 1011 Circular buffer size is 16 KB. 1100 Circular buffer size is 32 KB. 1101 Circular buffer size is 64 KB. 1110 Circular buffer size is 128 KB. 1111 Circular buffer size is 256 KB.
11–8 DMOD	Destination Address Modulo Defines the size of the destination data circular buffer used by the DMA Controller. If enabled (DMOD value is non-zero), the buffer base address is located on a boundary of the buffer size. The value of this

*Table continues on the next page...*

## DMA\_DCRn field descriptions (continued)

Field	Description
	<p>boundary depends on the initial destination address (DAR). The base address should be aligned to a 0-modulo-(circular buffer size) boundary. Misaligned buffers are not possible. The boundary is forced to the value determined by the upper address bits in the field selection.</p> <p>0000 Buffer disabled  0001 Circular buffer size is 16 bytes  0010 Circular buffer size is 32 bytes  0011 Circular buffer size is 64 bytes  0100 Circular buffer size is 128 bytes  0101 Circular buffer size is 256 bytes  0110 Circular buffer size is 512 bytes  0111 Circular buffer size is 1 KB  1000 Circular buffer size is 2 KB  1001 Circular buffer size is 4 KB  1010 Circular buffer size is 8 KB  1011 Circular buffer size is 16 KB  1100 Circular buffer size is 32 KB  1101 Circular buffer size is 64 KB  1110 Circular buffer size is 128 KB  1111 Circular buffer size is 256 KB</p>
7 D_REQ	<p>Disable Request</p> <p>DMA hardware automatically clears the corresponding DCRn[ERQ] bit when the byte count register reaches 0.</p> <p>0 ERQ bit is not affected.  1 ERQ bit is cleared when the BCR is exhausted.</p>
6 Reserved	<p>This field is reserved.  This read-only field is reserved and always has the value 0.</p>
5-4 LINKCC	<p>Link Channel Control</p> <p>Allows DMA channels to have their transfers linked. The current DMA channel triggers a DMA request to the linked channels (LCH1 or LCH2) depending on the condition described by the LINKCC bits.</p> <p>If not in cycle steal mode (DCRn[CS]=0) and LINKCC equals 01 or 10, no link to LCH1 occurs.</p> <p>If LINKCC equals 01, a link to LCH1 is created after each cycle-steal transfer performed by the current DMA channel is completed. As the last cycle-steal is performed and the BCR reaches zero, then the link to LCH1 is closed and a link to LCH2 is created.</p> <p>00 No channel-to-channel linking  01 Perform a link to channel LCH1 after each cycle-steal transfer followed by a link to LCH2 after the BCR decrements to 0.  10 Perform a link to channel LCH1 after each cycle-steal transfer  11 Perform a link to channel LCH1 after the BCR decrements to 0.</p>
3-2 LCH1	<p>Link Channel 1</p> <p>Indicates the DMA channel assigned as link channel 1. The link channel number cannot be the same as the currently executing channel, and generates a configuration error if this is attempted (DSRn[CE] is set).</p> <p>00 DMA Channel 0  01 DMA Channel 1</p>

Table continues on the next page...

DMA\_DCR $n$  field descriptions (continued)

Field	Description
	10 DMA Channel 2 11 DMA Channel 3
LCH2	Link Channel 2  Indicates the DMA channel assigned as link channel 2. The link channel number cannot be the same as the currently executing channel, and generates a configuration error if this is attempted (DSR $n$ [CE] is set).  00 DMA Channel 0 01 DMA Channel 1 10 DMA Channel 2 11 DMA Channel 3

## 21.4 Functional Description

In the following discussion, the term DMA request implies that DCR $n$ [START] is set, or DCR $n$ [ERQ] is set and then followed by assertion of the properly selected DMA peripheral request. DCR $n$ [START] is cleared when the channel is activated.

Before initiating a dual-address access, the DMA module verifies that DCR $n$ [SSIZE] and DCR $n$ [DSIZE] are consistent with the source and destination addresses. If they are not consistent, the configuration error bit, DSR $n$ [CE], is set. If misalignment is detected, no transfer occurs, DSR $n$ [CE] is set, and, depending on the DCR configuration, an interrupt event may be issued. If the auto-align bit, DCR $n$ [AA], is set, error checking is performed on the appropriate registers.

A read/write transfer sequence reads data from the source address and writes it to the destination address. The number of bytes transferred is the largest of the sizes specified by DCR $n$ [SSIZE] and DCR $n$ [DSIZE] in the DMA Control Registers (DCR $n$ ).

Source and destination address registers (SAR $n$  and DAR $n$ ) can be programmed in the DCR $n$  to increment at the completion of a successful transfer.

### 21.4.1 Transfer requests (Cycle-Steal and Continuous modes)

The DMA channel supports software-initiated or peripheral-initiated requests. A request is issued by setting DCR $n$ [START] or when the selected peripheral request asserts and DCR $n$ [ERQ] is set. Setting DCR $n$ [ERQ] enables recognition of the peripheral DMA requests. Selecting between cycle-steal and continuous modes minimizes bus usage for either type of request.

## Functional Description

- Cycle-steal mode ( $DCRn[CS] = 1$ )—Only one complete transfer from source to destination occurs for each request. If  $DCRn[ERQ]$  is set, the request is peripheral initiated. A software-initiated request is enabled by setting  $DCRn[START]$ .
- Continuous mode ( $DCRn[CS] = 0$ )—After a software-initiated or peripheral request, the DMA continuously transfers data until  $BCRn$  reaches 0. The DMA performs the specified number of transfers, then retires the channel.

In either mode, the crossbar switch performs independent arbitration on each slave port after each transaction.

### 21.4.2 Channel initialization and startup

Before a data transfer starts, the channel's transfer control descriptor must be initialized with information describing configuration, request-generation method, and pointers to the data to be moved.

#### 21.4.2.1 Channel prioritization

The four DMA channels are prioritized based on number, with channel 0 having highest priority and channel 3 having the lowest, that is, channel 0 > channel 1 > channel 2 > channel 3.

Simultaneous peripheral requests activate the channels based on this priority order. Once activated, a channel runs to completion as defined by  $DCRn[CS]$  and  $BCRn$ .

#### 21.4.2.2 Programming the DMA Controller Module

##### CAUTION

During a channel's execution, writes to programming model registers can corrupt the data transfer. The DMA module itself does not have a mechanism to prevent writes to registers during a channel's execution.

General guidelines for programming the DMA are:

- $TCDn$  is initialized.



- $SAR_n$  is loaded with the source (read) address. If the transfer is from a peripheral device to memory or to another peripheral, the source address is the location of the peripheral data register. If the transfer is from memory to a peripheral device or to memory, the source address is the starting address of the data block. This can be any appropriately aligned address.
- $DAR_n$  is initialized with the destination (write) address. If the transfer is from a peripheral device to memory, or from memory to memory,  $DAR_n$  is loaded with the starting address of the data block to be written. If the transfer is from memory to a peripheral device, or from a peripheral device to a peripheral device,  $DAR_n$  is loaded with the address of the peripheral data register. This address can be any appropriately aligned address.
- $SAR_n$  and  $DAR_n$  change after each data transfer depending on  $DCR_n[SSIZE, DSIZE, SINC, DINC, SMOD, DMOD]$  and the starting addresses. Increment values can be 1, 2, or 4 for 8-bit, 16-bit, or 32-bit transfers, respectively. If the address register is programmed to remain unchanged, the register is not incremented after the data transfer.
- $BCR_n[BCR]$  must be loaded with the total number of bytes to be transferred. It is decremented by 1, 2, or 4 at the end of each transfer, depending on the transfer size.  $DSR_n[DONE]$  must be cleared for channel startup.
- After the channel has been initialized, it may be started by setting  $DCR_n[START]$  or a properly selected peripheral DMA request, depending on the status of  $DCR_n[ERQ]$ . For a software-initiated transfer, the channel can be started by setting  $DCR_n[START]$  as part of a single 32-bit write to the last 32 bits of the  $TCD_n$ ; that is, it is not required to write the  $DCR_n$  with  $START$  cleared and then perform a second write to explicitly set  $START$ .
- Programming the channel for a software-initiated request causes the channel to request the system bus and start transferring data immediately. If the channel is programmed for peripheral-initiated request, a properly selected peripheral DMA request must be asserted before the channel begins the system bus transfers.
- The hardware can automatically clear  $DCR_n[ERQ]$ , disabling the peripheral request, when  $BCR_n$  reaches zero by setting  $DCR_n[D\_REQ]$ .
- Changes to  $DCR_n$  are effective immediately while the channel is active. To avoid problems with changing a DMA channel setup, write a one to  $DSR_n[DONE]$  to stop the DMA channel.

### 21.4.3 Dual-Address Data Transfer Mode

Each channel supports dual-address transfers. Dual-address transfers consist of a source data read and a destination data write. The DMA controller module begins a dual-address transfer sequence after a DMA request. If no error condition exists,  $DSR_n[REQ]$  is set.

- Dual-address read—The DMA controller drives the  $SAR_n$  value onto the system address bus. If  $DCR_n[SINC]$  is set, the  $SAR_n$  increments by the appropriate number of bytes upon a successful read cycle. When the appropriate number of read cycles complete (multiple reads if the destination size is larger than the source), the DMA initiates the write portion of the transfer.

If a termination error occurs,  $DSR_n[BES, DONE]$  are set and DMA transactions stop.

- Dual-address write—The DMA controller drives the  $DAR_n$  value onto the system address bus. When the appropriate number of write cycles complete (multiple writes if the source size is larger than the destination),  $DAR_n$  increments by the appropriate number of bytes if  $DCR_n[DINC]$  is set.  $BCR_n$  decrements by the appropriate number of bytes.  $DSR_n[DONE]$  is set when  $BCR_n$  reaches zero. If the  $BCR_n$  is greater than zero, another read/write transfer is initiated if continuous mode is enabled ( $DCR_n[CS] = 0$ ).

If a termination error occurs,  $DSR_n[BED, DONE]$  are set and DMA transactions stop.

### 21.4.4 Advanced Data Transfer Controls: Auto-Alignment

Typically, auto-alignment for DMA transfers applies for transfers of large blocks of data. As a result, it does not apply for peripheral-initiated cycle-steal transfers.

Auto-alignment allows block transfers to occur at the optimal size based on the address, byte count, and programmed size. To use this feature,  $DCR_n[AA]$  must be set. The source is auto-aligned if  $DCR_n[SSIZE]$  indicates a transfer size larger than  $DCR_n[DSIZE]$ . Source alignment takes precedence over the destination when the source and destination sizes are equal. Otherwise, the destination is auto-aligned. The address register chosen for alignment increments regardless of the increment value. Configuration error checking is performed on registers not chosen for alignment.

If  $BCR_n$  is greater than 16, the address determines transfer size. Transfers of 8 bits, 16 bits, or 32 bits are transferred until the address is aligned to the programmed size boundary, at which time accesses begin using the programmed size. If  $BCR_n$  is less than 16 at the start of a transfer, the number of bytes remaining dictates transfer size.

Consider this example:

- AA equals 1.
- SAR<sub>n</sub> equals 0x2000\_0001.
- BCR<sub>n</sub> equals 0x00\_00F0.
- SSIZE equals 00 (32 bits).
- DSIZE equals 01 (8 bits).

Because SSIZE > DSIZE, the source is auto-aligned. Error checking is performed on destination registers. The access sequence is as follows:

1. Read 1 byte from 0x2000\_0001, increment SAR<sub>n</sub>, write 1 byte (using DAR<sub>n</sub>).
2. Read 2 bytes from 0x2000\_0002, increment SAR<sub>n</sub>, write 2 bytes.
3. Read 4 bytes from 0x2000\_0004, increment SAR<sub>n</sub>, write 4 bytes.
4. Repeat 4-byte operations until SAR<sub>n</sub> equals 0x2000\_00F0.
5. Read byte from 0x2000\_00F0, increment SAR<sub>n</sub>, write byte.

If DSIZE is another size, data writes are optimized to write the largest size allowed based on the address, but not exceeding the configured size.

## 21.4.5 Termination

An unsuccessful transfer can terminate for one of the following reasons:

- Error conditions—When the DMA encounters a read or write cycle that terminates with an error condition, DSR<sub>n</sub>[BES] is set for a read and DSR<sub>n</sub>[BED] is set for a write before the transfer is halted. If the error occurred in a write cycle, data in the internal holding registers is lost.
- Interrupts—If DCR<sub>n</sub>[EINT] is set, the DMA drives the appropriate interrupt request signal. The processor can read DSR<sub>n</sub> to determine whether the transfer terminated successfully or with an error. DSR<sub>n</sub>[DONE] is then written with a 1 to clear the interrupt, DSR<sub>n</sub>[DONE], and error status bits.



# Chapter 22

## Reset Control Module (RCM)

### 22.1 Introduction

Information found here describes the registers of the Reset Control Module (RCM). The RCM implements many of the reset functions for the chip. See the chip's reset chapter for more information.

See [AN4503: Power Management for Kinetis MCUs](#) for further details on using the RCM.

### 22.2 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="#">22.2.1/318</a>
4007_F001	System Reset Status Register 1 (RCM_SRS1)	8	R	00h	<a href="#">22.2.2/319</a>
4007_F004	Reset Pin Filter Control register (RCM_RPFC)	8	R/W	00h	<a href="#">22.2.3/320</a>
4007_F005	Reset Pin Filter Width register (RCM_RPFW)	8	R/W	00h	<a href="#">22.2.4/321</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_F006	Force Mode Register (RCM_FM)	8	R/W	00h	<a href="#">22.2.5/323</a>
4007_F007	Mode Register (RCM_MR)	8	R/W	See section	<a href="#">22.2.6/323</a>
4007_F008	Sticky System Reset Status Register 0 (RCM_SSRS0)	8	R/W	82h	<a href="#">22.2.7/324</a>
4007_F009	Sticky System Reset Status Register 1 (RCM_SSRS1)	8	R/W	00h	<a href="#">22.2.8/325</a>

**22.2.1 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		0	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>

*Table continues on the next page...*

**RCM\_SRS0 field descriptions (continued)**

Field	Description
5 WDOG	<p>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> <p>0 Reset not caused by watchdog timeout 1 Reset caused by watchdog timeout</p>
4–3 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. This read-only field is reserved and always has the value 0.</p>
1 LVD	<p>Low-Voltage Detect Reset</p> <p>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.</p> <p>0 Reset not caused by LVD trip or POR 1 Reset caused by LVD trip or POR</p>
0 WAKEUP	<p>Low Leakage Wakeup Reset</p> <p>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.</p> <p>0 Reset not caused by LLWU module wakeup source 1 Reset caused by LLWU module wakeup source</p>

**22.2.2 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	0
Write								
Reset	0	0	0	0	0	0	0	0

### RCM\_SRS1 field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SACKERR	Stop Mode Acknowledge Error Reset  Indicates that after an attempt to enter Stop mode, a reset has been caused by a failure of one or more peripherals to acknowledge within approximately one second to enter stop mode.  0 Reset not caused by peripheral failure to acknowledge attempt to enter stop mode 1 Reset caused by peripheral failure to acknowledge attempt to enter stop mode
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 MDM_AP	MDM-AP System Reset Request  Indicates a reset has been caused by the host debugger system setting of the System Reset Request bit in the MDM-AP Control Register.  0 Reset not caused by host debugger system setting of the System Reset Request bit 1 Reset caused by host debugger system setting of the System Reset Request bit
2 SW	Software  Indicates a reset has been caused by software setting of SYSRESETREQ bit in Application Interrupt and Reset Control Register in the ARM core.  0 Reset not caused by software setting of SYSRESETREQ bit 1 Reset caused by software setting of SYSRESETREQ bit
1 LOCKUP	Core Lockup  Indicates a reset has been caused by the ARM core indication of a LOCKUP event.  0 Reset not caused by core LOCKUP event 1 Reset caused by core LOCKUP event
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

## 22.2.3 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
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

**22.2.4 Reset Pin Filter Width register (RCM\_RPFW)****NOTE**

The reset values of the bits in the RSTFLTSEL field are for Chip POR only. They are unaffected by other reset types.

Address: 4007\_F000h base + 5h offset = 4007\_F005h

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

**RCM\_RPFW field descriptions**

Field	Description
7-5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RSTFLTSEL	<p>Reset Pin Filter Bus Clock Select</p> <p>Selects the reset pin bus clock filter width.</p> <p>00000 Bus clock filter count is 1            00001 Bus clock filter count is 2            00010 Bus clock filter count is 3            00011 Bus clock filter count is 4            00100 Bus clock filter count is 5            00101 Bus clock filter count is 6            00110 Bus clock filter count is 7            00111 Bus clock filter count is 8            01000 Bus clock filter count is 9            01001 Bus clock filter count is 10            01010 Bus clock filter count is 11            01011 Bus clock filter count is 12            01100 Bus clock filter count is 13            01101 Bus clock filter count is 14            01110 Bus clock filter count is 15            01111 Bus clock filter count is 16            10000 Bus clock filter count is 17            10001 Bus clock filter count is 18            10010 Bus clock filter count is 19            10011 Bus clock filter count is 20            10100 Bus clock filter count is 21            10101 Bus clock filter count is 22            10110 Bus clock filter count is 23            10111 Bus clock filter count is 24            11000 Bus clock filter count is 25            11001 Bus clock filter count is 26            11010 Bus clock filter count is 27            11011 Bus clock filter count is 28            11100 Bus clock filter count is 29            11101 Bus clock filter count is 30            11110 Bus clock filter count is 31            11111 Bus clock filter count is 32</p>

## 22.2.5 Force Mode Register (RCM\_FM)

### NOTE

The reset values of the bits in the FORCEROM field are for Chip POR only. They are unaffected by other reset types.

Address: 4007\_F000h base + 6h offset = 4007\_F006h

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

### RCM\_FM field descriptions

Field	Description
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–1 FORCEROM	Force ROM Boot When either bit is set, will force boot from ROM during all subsequent system resets.  00 No effect 01 Force boot from ROM with RCM_MR[1] set. 10 Force boot from ROM with RCM_MR[2] set. 11 Force boot from ROM with RCM_MR[2:1] set.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

## 22.2.6 Mode Register (RCM\_MR)

This register includes status flags to indicate the state of the mode pins during the last Chip Reset.

Address: 4007\_F000h base + 7h offset = 4007\_F007h

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

\* Notes:

- BOOTROM field: The reset state of this register depends on the boot mode.

**RCM\_MR field descriptions**

Field	Description
7-3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2-1 BOOTROM	<p>Boot ROM Configuration</p> <p>Indicates the boot source, the boot source remains set until the next System Reset or software can write logic one to clear the corresponding mode bit.</p> <p>While either bit is set, the NMI input is disabled and the vector table is relocated to the ROM base address at 0x1C00_0000. These bits should be cleared by writing logic one before executing any code from either Flash or SRAM.</p> <p>00 Boot from Flash                      01 Boot from ROM due to BOOTCFG0 pin assertion                      10 Boot form ROM due to FOPT[7] configuration                      11 Boot from ROM due to both BOOTCFG0 pin assertion and FOPT[7] configuration</p>
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

**22.2.7 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		0	SLVD	SWAKEUP
Write	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>

*Table continues on the next page...*

## RCM\_SSRS0 field descriptions (continued)

Field	Description
	0 Reset not caused by external reset pin 1 Reset caused by external reset pin
5 SWDOG	Sticky Watchdog  Indicates a reset has been caused by the watchdog timer timing out. This reset source can be blocked by disabling the watchdog.  0 Reset not caused by watchdog timeout 1 Reset caused by watchdog timeout
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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 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.  0 Reset not caused by LLWU module wakeup source 1 Reset caused by LLWU module wakeup source

## 22.2.8 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	0
Write			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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

# Chapter 23

## Analog-to-Digital Converter (ADC)

### 23.1 Chip-specific ADC information

#### 23.1.1 ADC instantiation information

This device contains one 16-bit successive approximation ADC with up to 16 channels.

The ADC supports both software and hardware triggers. The hardware trigger sources are listed in the [Module-to-Module section](#).

The number of ADC channels present on the device is determined by the pinout of the specific device package and is shown in the following table.

**Table 23-1. Number of KL13 ADC channels and ADC pins (includes both SExa andSExb)**

Device	Number of ADC Channels	Number of ADC Pins	Package
MKL13Z32VLH4 MKL13Z32VLH4	20	20	64LQFP
MKL13Z32VLK4 MKL13Z64VLK4	20	20	80LQFP

#### 23.1.2 DMA Support on ADC

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

## 23.1.3 ADC0 connections/channel assignment

### NOTE

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

**Table 23-2. ADC0 channel assignment**

ADC channel (SC1n[ADCH])	Channel	Input signal (SC1n[DIFF]= 1)	Input signal (SC1n[DIFF]= 0)
00000	DAD0	ADC0_DP0 and ADC0_DM0	ADC0_DP0/ADC0_SE0
00001	DAD1	ADC0_DP1 and ADC0_DM1	ADC0_DP1/ADC0_SE1
00010	DAD2	ADC0_DP2 and ADC0_DM2	ADC0_DP2/ADC0_SE2
00011	DAD3	ADC0_DP3 and ADC0_DM3	ADC0_DP3/ADC0_SE3
00100 <sup>1</sup>	AD4a	Reserved	ADC0_DM0/ADC0_SE4a
00101 <sup>1</sup>	AD5a	Reserved	ADC0_DM1/ADC0_SE5a
00110 <sup>1</sup>	AD6a	Reserved	ADC0_DM2/ADC0_SE6a
00111 <sup>1</sup>	AD7a	Reserved	ADC0_DM3/ADC0_SE7a
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	ADC0_SE11
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	Reserved
10010	AD18	Reserved	Reserved
10011	AD19	Reserved	Reserved
10100	AD20	Reserved	Reserved
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
11010	AD26	Temperature Sensor (Diff)	Temperature Sensor (S.E)
11011	AD27	Bandgap (Diff) <sup>2</sup>	Bandgap (S.E) <sup>2</sup>

Table continues on the next page...



**Table 23-2. ADC0 channel assignment (continued)**

ADC channel (SC1n[ADCH])	Channel	Input signal (SC1n[DIF]=1)	Input signal (SC1n[DIF]=0)
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. This is the PMC bandgap 1V reference voltage. Prior to reading from this ADC channel, ensure that you enable the bandgap buffer by setting the PMC\_REGSC[BGBE] bit. Refer to the device data sheet for the bandgap voltage ( $V_{BG}$ ) specification.

### 23.1.4 ADC analog supply and reference connections

This device includes dedicated VDDA and VSSA pins.

This device contains dedicated VREFH and VREFL pins in 32 pin above packages. It also includes dedicated VDDA and VSSA pins on all packages.

The output of On-chip 1.2V high precision voltage reference VREF\_OUT shares with VREFH on all packages. When VREF\_OUT is enabled, this pin needs to connect a capacitor to ground.

### 23.1.5 Alternate clock

For this device, the alternate clock is connected to the external reference clock (OSCERCLK).

#### NOTE

This clock 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.

## 23.2 Introduction

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.

### 23.2.1 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

## 23.2.2 Block diagram

The following figure is the ADC module block diagram.

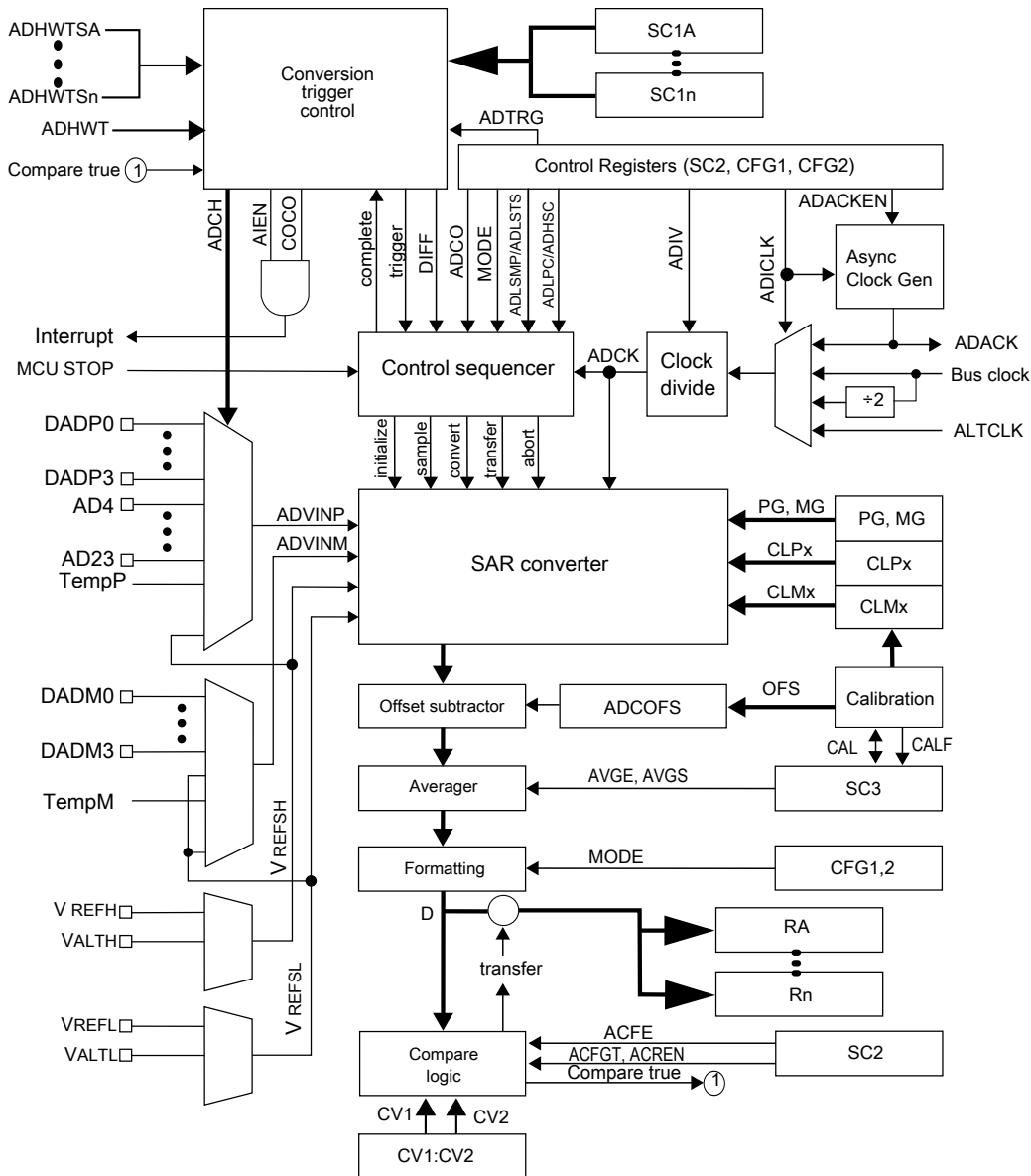


Figure 23-1. ADC block diagram

## 23.3 ADC signal descriptions

The ADC module supports up to 4 pairs of differential inputs and up to 24 single-ended inputs.

Each differential pair requires two inputs, DADPx and DADMx. The ADC also requires four supply/reference/ground connections.

**NOTE**

For the number of channels supported on this device as well as information regarding other chip-specific inputs into the ADC block, see the chip-specific ADC configuration information.

**Table 23-3. 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

**23.3.1 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.

**23.3.2 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>.

**23.3.3 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>ALTL</sub>). These voltage references are selected using SC2[REFSEL]. The alternate V<sub>ALTH</sub> and

$V_{ALTL}$  voltage reference pair may select additional external pins or internal sources depending on MCU configuration. See the chip configuration information on the Voltage References specific to this MCU.

In some packages,  $V_{REFH}$  is connected in the package to  $V_{DDA}$  and  $V_{REFL}$  to  $V_{SSA}$ . If externally available, the positive reference(s) may be connected to the same potential as  $V_{DDA}$  or may be driven by an external source to a level between the minimum Ref Voltage High and the  $V_{DDA}$  potential.  $V_{REFH}$  must never exceed  $V_{DDA}$ . Connect the ground references to the same voltage potential as  $V_{SSA}$ .

### 23.3.4 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.

### 23.3.5 Differential Analog Channel Inputs (DADx)

The ADC module supports up to four differential analog channel inputs. Each differential analog input is a pair of external pins,  $DADPx$  and  $DADMx$ , referenced to each other to provide the most accurate analog to digital readings. A differential input is selected for conversion through  $SC1[ADCH]$  when  $SC1n[DIFF]$  is high. All  $DADPx$  inputs may be used as single-ended inputs if  $SC1n[DIFF]$  is low. In certain MCU configurations, some  $DADMx$  inputs may also be used as single-ended inputs if  $SC1n[DIFF]$  is low. For ADC connections specific to this device, see the chip-specific ADC information.

## 23.4 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="#">23.4.1/334</a>
4003_B004	ADC Status and Control Registers 1 (ADC0_SC1B)	32	R/W	0000_001Fh	<a href="#">23.4.1/334</a>
4003_B008	ADC Configuration Register 1 (ADC0_CFG1)	32	R/W	0000_0000h	<a href="#">23.4.2/338</a>
4003_B00C	ADC Configuration Register 2 (ADC0_CFG2)	32	R/W	0000_0000h	<a href="#">23.4.3/339</a>
4003_B010	ADC Data Result Register (ADC0_RA)	32	R	0000_0000h	<a href="#">23.4.4/340</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_B014	ADC Data Result Register (ADC0_RB)	32	R	0000_0000h	<a href="#">23.4.4/340</a>
4003_B018	Compare Value Registers (ADC0_CV1)	32	R/W	0000_0000h	<a href="#">23.4.5/342</a>
4003_B01C	Compare Value Registers (ADC0_CV2)	32	R/W	0000_0000h	<a href="#">23.4.5/342</a>
4003_B020	Status and Control Register 2 (ADC0_SC2)	32	R/W	0000_0000h	<a href="#">23.4.6/343</a>
4003_B024	Status and Control Register 3 (ADC0_SC3)	32	R/W	0000_0000h	<a href="#">23.4.7/345</a>
4003_B028	ADC Offset Correction Register (ADC0_OFS)	32	R/W	0000_0004h	<a href="#">23.4.8/346</a>
4003_B02C	ADC Plus-Side Gain Register (ADC0_PG)	32	R/W	0000_8200h	<a href="#">23.4.9/347</a>
4003_B030	ADC Minus-Side Gain Register (ADC0_MG)	32	R/W	0000_8200h	<a href="#">23.4.10/347</a>
4003_B034	ADC Plus-Side General Calibration Value Register (ADC0_CLPD)	32	R/W	0000_000Ah	<a href="#">23.4.11/348</a>
4003_B038	ADC Plus-Side General Calibration Value Register (ADC0_CLPS)	32	R/W	0000_0020h	<a href="#">23.4.12/349</a>
4003_B03C	ADC Plus-Side General Calibration Value Register (ADC0_CLP4)	32	R/W	0000_0200h	<a href="#">23.4.13/349</a>
4003_B040	ADC Plus-Side General Calibration Value Register (ADC0_CLP3)	32	R/W	0000_0100h	<a href="#">23.4.14/350</a>
4003_B044	ADC Plus-Side General Calibration Value Register (ADC0_CLP2)	32	R/W	0000_0080h	<a href="#">23.4.15/350</a>
4003_B048	ADC Plus-Side General Calibration Value Register (ADC0_CLP1)	32	R/W	0000_0040h	<a href="#">23.4.16/351</a>
4003_B04C	ADC Plus-Side General Calibration Value Register (ADC0_CLP0)	32	R/W	0000_0020h	<a href="#">23.4.17/351</a>
4003_B054	ADC Minus-Side General Calibration Value Register (ADC0_CLMD)	32	R/W	0000_000Ah	<a href="#">23.4.18/352</a>
4003_B058	ADC Minus-Side General Calibration Value Register (ADC0_CLMS)	32	R/W	0000_0020h	<a href="#">23.4.19/352</a>
4003_B05C	ADC Minus-Side General Calibration Value Register (ADC0_CLM4)	32	R/W	0000_0200h	<a href="#">23.4.20/353</a>
4003_B060	ADC Minus-Side General Calibration Value Register (ADC0_CLM3)	32	R/W	0000_0100h	<a href="#">23.4.21/353</a>
4003_B064	ADC Minus-Side General Calibration Value Register (ADC0_CLM2)	32	R/W	0000_0080h	<a href="#">23.4.22/354</a>
4003_B068	ADC Minus-Side General Calibration Value Register (ADC0_CLM1)	32	R/W	0000_0040h	<a href="#">23.4.23/354</a>
4003_B06C	ADC Minus-Side General Calibration Value Register (ADC0_CLM0)	32	R/W	0000_0020h	<a href="#">23.4.24/355</a>

### 23.4.1 ADC Status and Control Registers 1 (ADCx\_SC1n)

SC1A is used for both software and hardware trigger modes of operation.

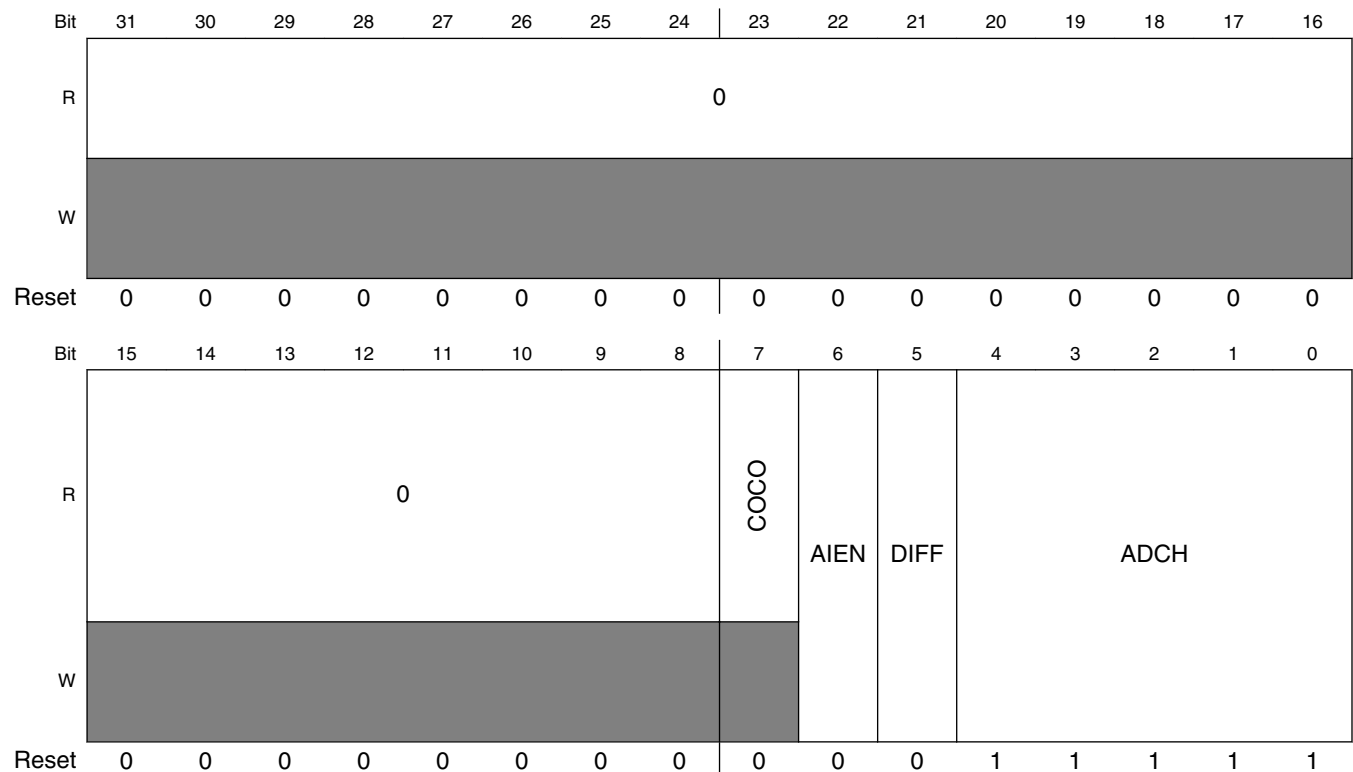
To allow sequential conversions of the ADC to be triggered by internal peripherals, the ADC can have more than one status and control register: one for each conversion. The SC1B–SC1n registers indicate potentially multiple SC1 registers for use only in hardware trigger mode. See the chip configuration 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 (module disabled).

Writing any of the SC1n registers while that specific SC1n register is actively controlling a conversion aborts the current conversion. None of the SC1B–SC1n registers are used for software trigger operation and therefore writes to the SC1B–SC1n registers do not initiate a new conversion.

Address: 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	<p>Conversion Complete Flag</p> <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>
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.</p>

Table continues on the next page...



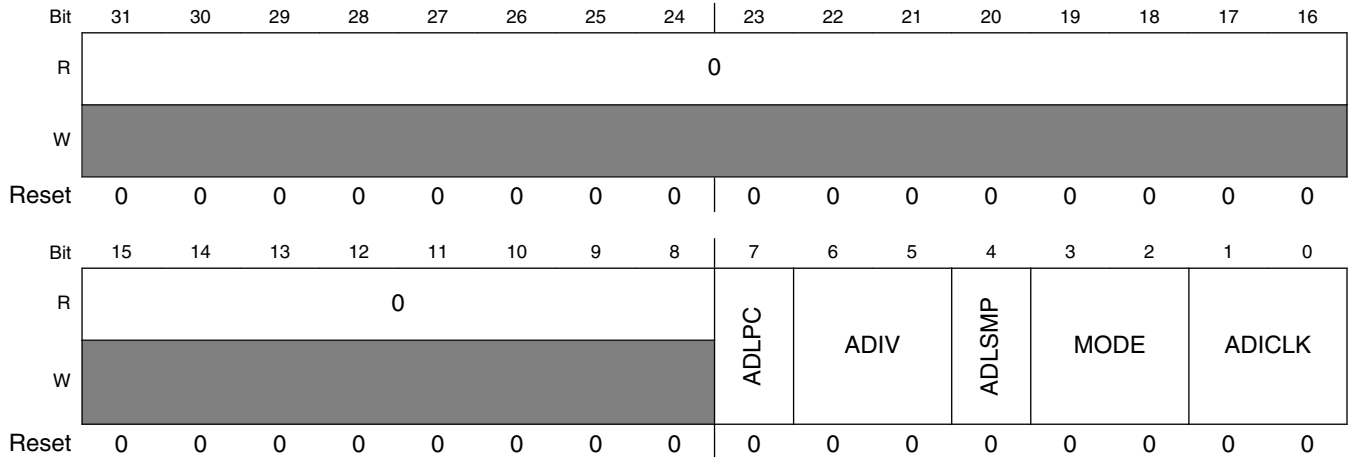
## ADCx\_SC1n field descriptions (continued)

Field	Description
01000	When DIFF=0, AD8 is selected as input; when DIFF=1, it is reserved.
01001	When DIFF=0, AD9 is selected as input; when DIFF=1, it is reserved.
01010	When DIFF=0, AD10 is selected as input; when DIFF=1, it is reserved.
01011	When DIFF=0, AD11 is selected as input; when DIFF=1, it is reserved.
01100	When DIFF=0, AD12 is selected as input; when DIFF=1, it is reserved.
01101	When DIFF=0, AD13 is selected as input; when DIFF=1, it is reserved.
01110	When DIFF=0, AD14 is selected as input; when DIFF=1, it is reserved.
01111	When DIFF=0, AD15 is selected as input; when DIFF=1, it is reserved.
10000	When DIFF=0, AD16 is selected as input; when DIFF=1, it is reserved.
10001	When DIFF=0, AD17 is selected as input; when DIFF=1, it is reserved.
10010	When DIFF=0, AD18 is selected as input; when DIFF=1, it is reserved.
10011	When DIFF=0, AD19 is selected as input; when DIFF=1, it is reserved.
10100	When DIFF=0, AD20 is selected as input; when DIFF=1, it is reserved.
10101	When DIFF=0, AD21 is selected as input; when DIFF=1, it is reserved.
10110	When DIFF=0, AD22 is selected as input; when DIFF=1, it is reserved.
10111	When DIFF=0, AD23 is selected as input; when DIFF=1, it is reserved.
11000	Reserved.
11001	Reserved.
11010	When DIFF=0, Temp Sensor (single-ended) is selected as input; when DIFF=1, Temp Sensor (differential) is selected as input.
11011	When DIFF=0, Bandgap (single-ended) is selected as input; when DIFF=1, Bandgap (differential) is selected as input.
11100	Reserved.
11101	When DIFF=0, $V_{REFSH}$ is selected as input; when DIFF=1, $-V_{REFSH}$ (differential) is selected as input. Voltage reference selected is determined by SC2[REFSEL].
11110	When DIFF=0, $V_{REFSL}$ is selected as input; when DIFF=1, it is reserved. Voltage reference selected is determined by SC2[REFSEL].
11111	Module is disabled.

### 23.4.2 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



#### ADCx\_CFG1 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADLPC	Low-Power Configuration  Controls the power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required.  0 Normal power configuration. 1 Low-power configuration. The power is reduced at the expense of maximum clock speed.
6–5 ADIV	Clock Divide Select  Selects the divide ratio used by the ADC to generate the internal clock ADCK.  00 The divide ratio is 1 and the clock rate is input clock. 01 The divide ratio is 2 and the clock rate is (input clock)/2. 10 The divide ratio is 4 and the clock rate is (input clock)/4. 11 The divide ratio is 8 and the clock rate is (input clock)/8.
4 ADLSMP	Sample Time Configuration  Selects between different sample times based on the conversion mode selected. This field adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption if continuous conversions are enabled and high conversion rates are not required. When ADLSMP=1, the long sample time select bits, (ADLSTS[1:0]), can select the extent of the long sample time.

Table continues on the next page...

## ADCx\_CFG1 field descriptions (continued)

Field	Description
	0 Short sample time. 1 Long sample time.
3–2 MODE	Conversion mode selection  Selects the ADC resolution mode.  00 When DIFF=0:It is single-ended 8-bit conversion; when DIFF=1, it is differential 9-bit conversion with 2's complement output. 01 When DIFF=0:It is single-ended 12-bit conversion ; when DIFF=1, it is differential 13-bit conversion with 2's complement output. 10 When DIFF=0:It is single-ended 10-bit conversion. ; when DIFF=1, it is differential 11-bit conversion with 2's complement output 11 When DIFF=0:It is single-ended 16-bit conversion..; when DIFF=1, it is differential 16-bit conversion with 2's complement output
ADICLK	Input Clock Select  Selects the input clock source to generate the internal clock, ADCK. Note that when the ADACK clock source is selected, it is not required to be active prior to conversion start. When it is selected and it is not active prior to a conversion start, when CFG2[ADACKEN]=0, the asynchronous clock is activated at the start of a conversion and deactivated when conversions are terminated. In this case, there is an associated clock startup delay each time the clock source is re-activated.  00 Bus clock 01 Bus clock divided by 2(BUSCLK/2) 10 Alternate clock (ALTCLK) 11 Asynchronous clock (ADACK)

## 23.4.3 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																
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																
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  Configures the ADC for very high-speed operation. The conversion sequence is altered with 2 ADCK cycles added to the conversion time to allow higher speed conversion clocks.  0 Normal conversion sequence selected. 1 High-speed conversion sequence selected with 2 additional ADCK cycles to total conversion time.
ADLSTS	Long Sample Time Select  Selects between the extended sample times when long sample time is selected, that is, when CFG1[ADLSMP]=1. This allows higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required.  00 Default longest sample time; 20 extra ADCK cycles; 24 ADCK cycles total. 01 12 extra ADCK cycles; 16 ADCK cycles total sample time. 10 6 extra ADCK cycles; 10 ADCK cycles total sample time. 11 2 extra ADCK cycles; 6 ADCK cycles total sample time.

### 23.4.4 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<sub>n</sub> 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 23-4. Data result register description**

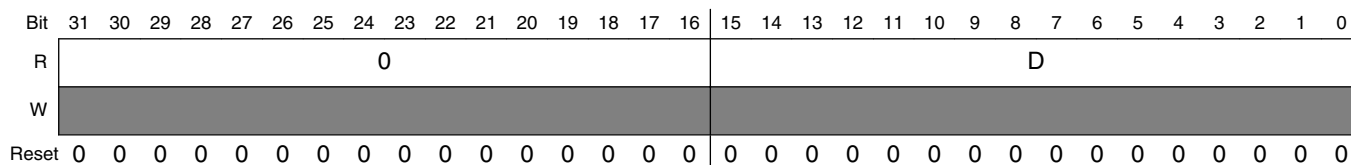
Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format
16-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Signed 2's complement
16-bit single-ended	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Unsigned right justified
13-bit differential	S	S	S	S	D	D	D	D	D	D	D	D	D	D	D	D	Sign-extended 2's complement
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
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



**ADCx\_Rn field descriptions**

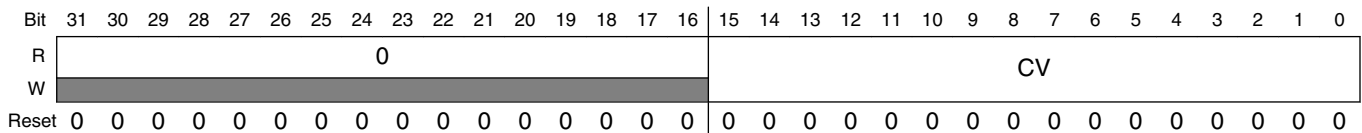
Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
D	Data result

### 23.4.5 Compare Value Registers (ADCx\_CVn)

The Compare Value Registers (CV1 and CV2) contain a compare value used to compare the conversion result when the compare function is enabled, that is, SC2[ACFE]=1. This register is formatted in the same way as the Rn registers in different modes of operation for both bit position definition and value format using unsigned or sign-extended 2's complement. Therefore, the compare function uses only the CVn fields that are related to the ADC mode of operation.

The compare value 2 register (CV2) is used only when the compare range function is enabled, that is, SC2[ACREN]=1.

Address: 4003\_B000h base + 18h offset + (4d × i), where i=0d to 1d



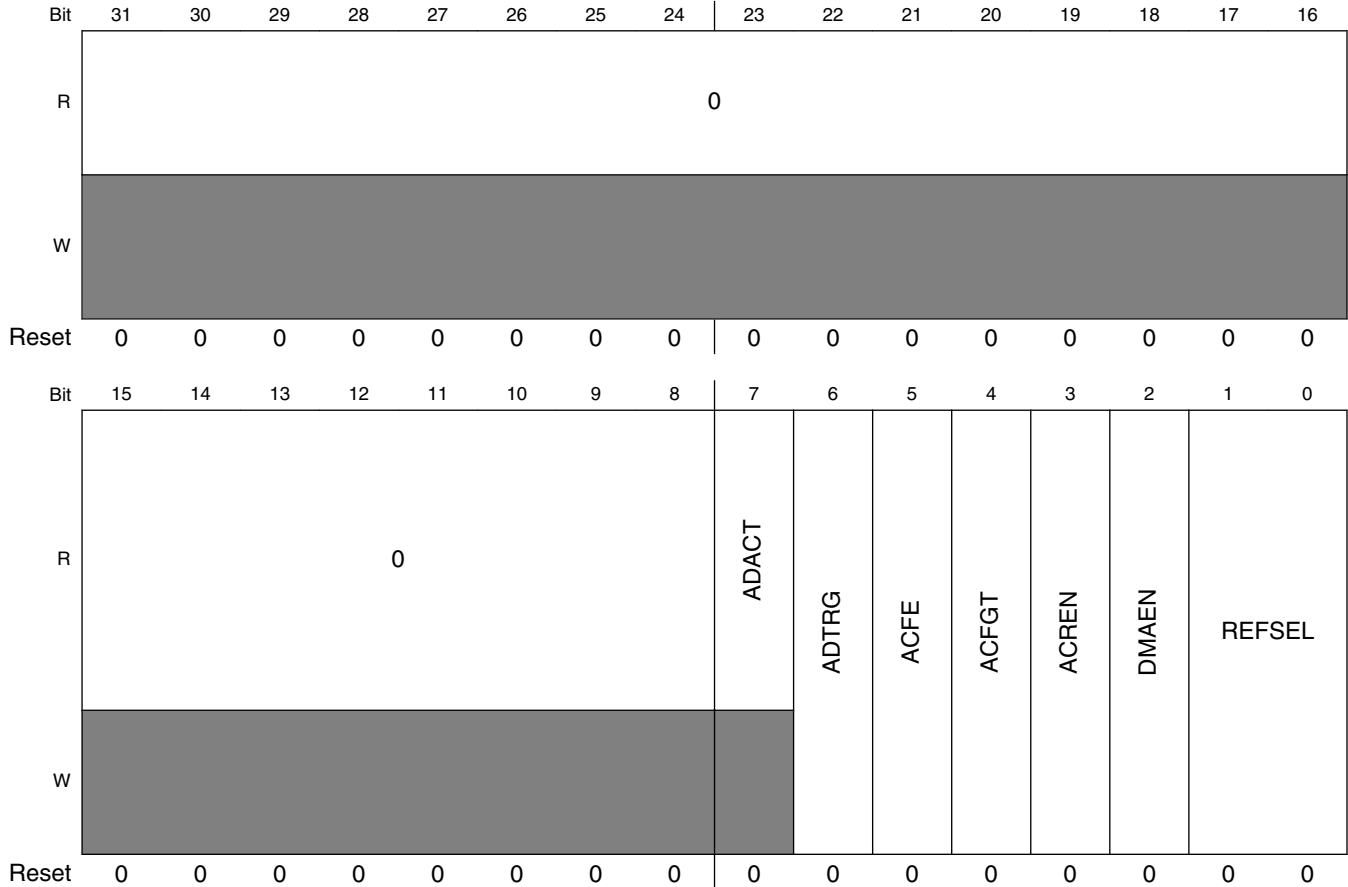
#### ADCx\_CVn field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CV	Compare Value.

### 23.4.6 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:

*Table continues on the next page...*

## ADCx\_SC2 field descriptions (continued)

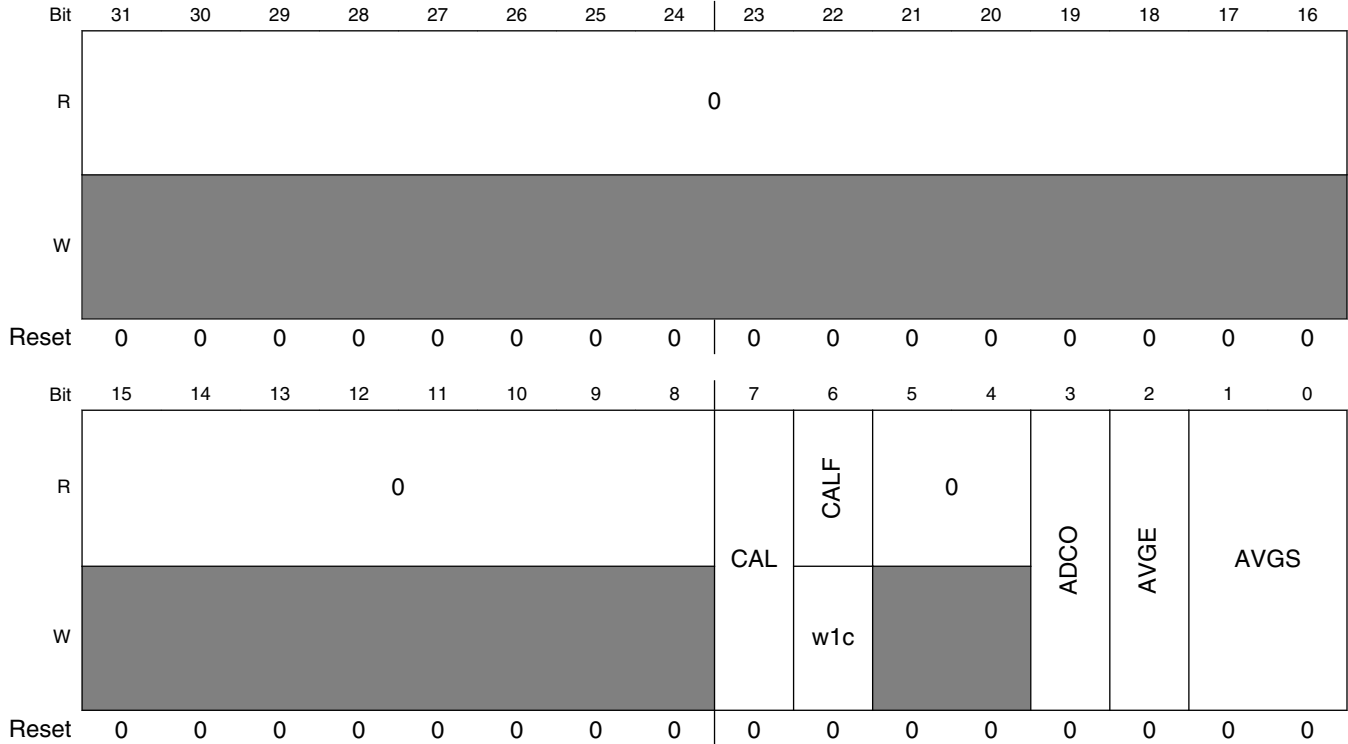
Field	Description
	<ul style="list-style-type: none"> <li>Software trigger: When software trigger is selected, a conversion is initiated following a write to SC1A.</li> <li>Hardware trigger: When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input after a pulse of the ADHWTSn input.</li> </ul> <p>0 Software trigger selected. 1 Hardware trigger selected.</p>
5 ACFE	<p>Compare Function Enable</p> <p>Enables the compare function.</p> <p>0 Compare function disabled. 1 Compare function enabled.</p>
4 ACFGT	<p>Compare Function Greater Than Enable</p> <p>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.</p> <p>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.</p>
3 ACREN	<p>Compare Function Range Enable</p> <p>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.</p> <p>0 Range function disabled. Only CV1 is compared. 1 Range function enabled. Both CV1 and CV2 are compared.</p>
2 DMAEN	<p>DMA Enable</p> <p>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.</p>
REFSEL	<p>Voltage Reference Selection</p> <p>Selects the voltage reference source used for conversions.</p> <p>00 Default voltage reference pin pair, that is, external pins V<sub>REFH</sub> and V<sub>REFL</sub> 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</p>



### 23.4.7 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.
7 CAL	Calibration  Begins the calibration sequence when set. This field stays set while the calibration is in progress and is cleared when the calibration sequence is completed. CALF must be checked to determine the result of the calibration sequence. Once started, the calibration routine cannot be interrupted by writes to the ADC registers or the results will be invalid and CALF will set. Setting CAL will abort any current conversion.
6 CALF	Calibration Failed Flag  Displays the result of the calibration sequence. The calibration sequence will fail if SC2[ADTRG] = 1, any ADC register is written, or any stop mode is entered before the calibration sequence completes. Writing 1 to CALF clears it.  0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.

Table continues on the next page...

**ADCx\_SC3 field descriptions (continued)**

Field	Description
5-4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 ADCO	Continuous Conversion Enable Enables continuous conversions.  0 One conversion or one set of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion. 1 Continuous conversions or sets of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion.
2 AVGE	Hardware Average Enable Enables the hardware average function of the ADC.  0 Hardware average function disabled. 1 Hardware average function enabled.
AVGS	Hardware Average Select Determines how many ADC conversions will be averaged to create the ADC average result.  00 4 samples averaged. 01 8 samples averaged. 10 16 samples averaged. 11 32 samples averaged.

**23.4.8 ADC Offset Correction Register (ADCx\_OFS)**

The ADC Offset Correction Register (OFS) contains the user-selected or calibration-generated offset error correction value. This register is a 2’s complement, left-justified, 16-bit value . The value in OFS is subtracted from the conversion and the result is transferred into the result registers, Rn. If the result is greater than the maximum or less than the minimum result value, it is forced to the appropriate limit for the current mode of operation.

For more information regarding the calibration procedure, please refer to the [Calibration function](#) section.

Address: 4003\_B000h base + 28h offset = 4003\_B028h

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

**ADCx\_OFS field descriptions**

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
OFS	Offset Error Correction Value

**23.4.9 ADC Plus-Side Gain Register (ADCx\_PG)**

The Plus-Side Gain Register (PG) contains the gain error correction for the plus-side input in differential mode or the overall conversion in single-ended mode. PG, a 16-bit real number in binary format, is the gain adjustment factor, with the radix point fixed between PG[15] and PG[14]. This register must be written by the user with the value described in the calibration procedure. Otherwise, the gain error specifications may not be met.

For more information regarding the calibration procedure, please refer to the [Calibration function](#) section.

Address: 4003\_B000h base + 2Ch offset = 4003\_B02Ch

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

**ADCx\_PG field descriptions**

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
PG	Plus-Side Gain

**23.4.10 ADC Minus-Side Gain Register (ADCx\_MG)**

The Minus-Side Gain Register (MG) contains the gain error correction for the minus-side input in differential mode. This register is ignored in single-ended mode. MG, a 16-bit real number in binary format, is the gain adjustment factor, with the radix point fixed between MG[15] and MG[14]. This register must be written by the user with the value described in the calibration procedure. Otherwise, the gain error specifications may not be met.

## Memory map and register definitions

For more information regarding the calibration procedure, please refer to the [Calibration function](#) section.

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.
MG	Minus-Side Gain

## 23.4.11 ADC Plus-Side General Calibration Value Register (ADCx\_CLPD)

The Plus-Side General Calibration Value Registers (CLPx) contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLP0[5:0], CLP1[6:0], CLP2[7:0], CLP3[8:0], CLP4[9:0], CLPS[5:0], and CLPD[5:0]. CLPx are automatically set when the self-calibration sequence is done, that is, CAL is cleared. If these registers are written by the user after calibration, the linearity error specifications may not be met.

For more information regarding the calibration procedure, please refer to the [Calibration function](#) section.

Address: 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.
CLPD	Calibration Value Calibration Value

### 23.4.12 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																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

#### ADCx\_CLPS field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLPS	Calibration Value Calibration Value

### 23.4.13 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																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

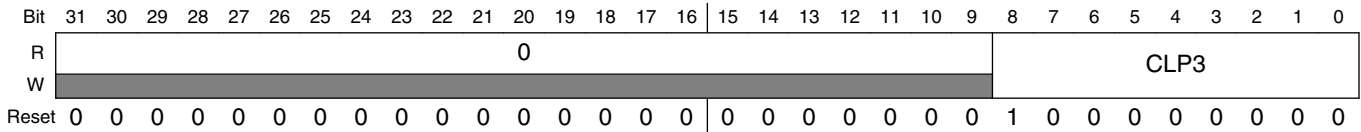
#### ADCx\_CLP4 field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP4	Calibration Value Calibration Value

### 23.4.14 ADC Plus-Side General Calibration Value Register (ADCx\_CLP3)

For more information, see CLPD register description.

Address: 4003\_B000h base + 40h offset = 4003\_B040h



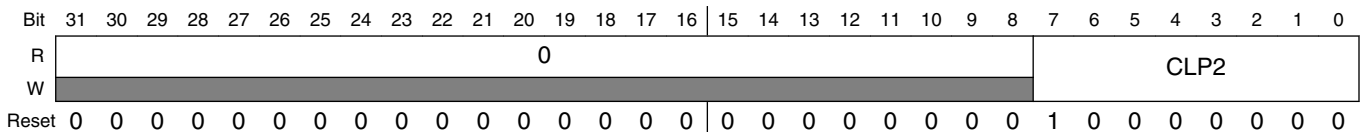
#### ADCx\_CLP3 field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP3	Calibration Value Calibration Value

### 23.4.15 ADC Plus-Side General Calibration Value Register (ADCx\_CLP2)

For more information, see CLPD register description.

Address: 4003\_B000h base + 44h offset = 4003\_B044h



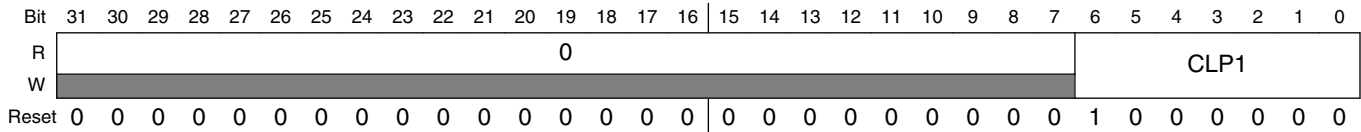
#### ADCx\_CLP2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP2	Calibration Value Calibration Value

### 23.4.16 ADC Plus-Side General Calibration Value Register (ADCx\_CLP1)

For more information, see CLPD register description.

Address: 4003\_B000h base + 48h offset = 4003\_B048h



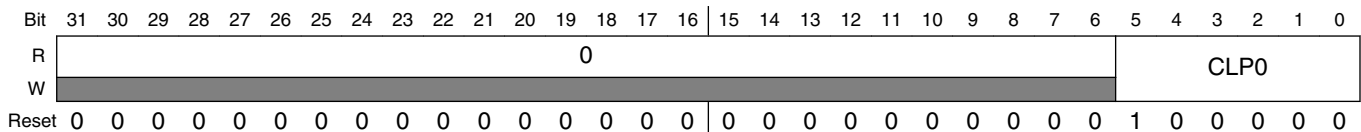
#### ADCx\_CLP1 field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP1	Calibration Value Calibration Value

### 23.4.17 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

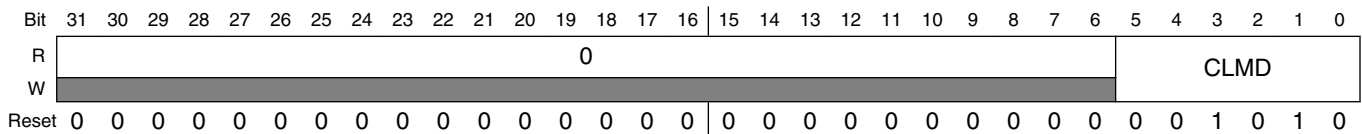
Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLP0	Calibration Value Calibration Value

### 23.4.18 ADC Minus-Side General Calibration Value Register (ADCx\_CLMD)

The Minus-Side General Calibration Value (CLMx) registers contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLM0[5:0], CLM1[6:0], CLM2[7:0], CLM3[8:0], CLM4[9:0], CLMS[5:0], and CLMD[5:0]. CLMx are automatically set when the self-calibration sequence is done, that is, CAL is cleared. If these registers are written by the user after calibration, the linearity error specifications may not be met.

For more information regarding the calibration procedure, please refer to the [Calibration function](#) section.

Address: 4003\_B000h base + 54h offset = 4003\_B054h



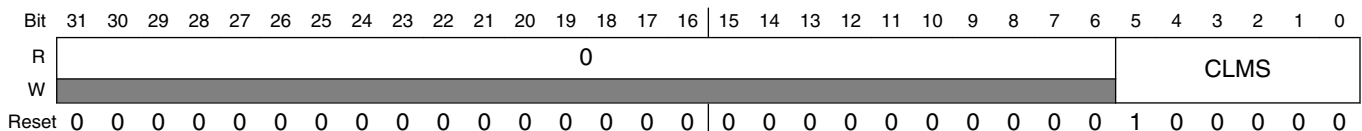
#### ADCx\_CLMD field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLMD	Calibration Value Calibration Value

### 23.4.19 ADC Minus-Side General Calibration Value Register (ADCx\_CLMS)

For more information, see CLMD register description.

Address: 4003\_B000h base + 58h offset = 4003\_B058h





**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.
CLMS	Calibration Value Calibration Value

**23.4.20 ADC Minus-Side General Calibration Value Register (ADCx\_CLM4)**

For more information, see CLMD register description.

Address: 4003\_B000h base + 5Ch offset = 4003\_B05Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																CLM4															
W	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\_CLM4 field descriptions**

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM4	Calibration Value Calibration Value

**23.4.21 ADC Minus-Side General Calibration Value Register (ADCx\_CLM3)**

For more information, see CLMD register description.

Address: 4003\_B000h base + 60h offset = 4003\_B060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																CLM3															
W	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	1	0	0	0	0	0	0	0	0

**ADCx\_CLM3 field descriptions**

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

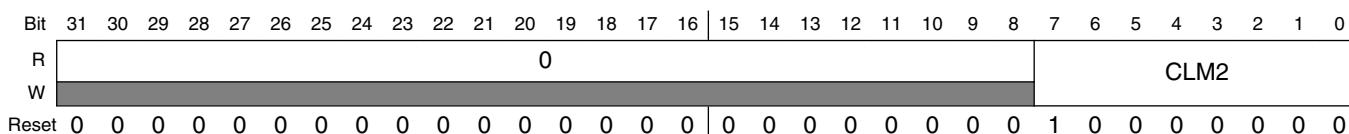
**ADCx\_CLM3 field descriptions (continued)**

Field	Description
CLM3	Calibration Value
	Calibration Value

**23.4.22 ADC Minus-Side General Calibration Value Register (ADCx\_CLM2)**

For more information, see CLMD register description.

Address: 4003\_B000h base + 64h offset = 4003\_B064h



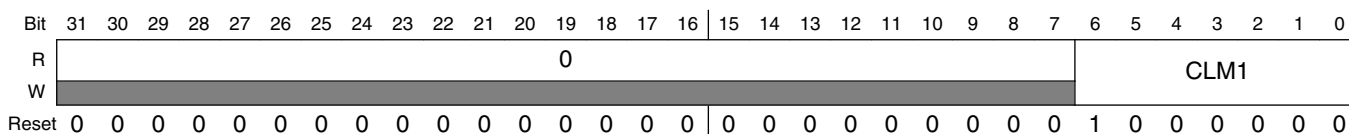
**ADCx\_CLM2 field descriptions**

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM2	Calibration Value
	Calibration Value

**23.4.23 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.
CLM1	Calibration Value
	Calibration Value

### 23.4.24 ADC Minus-Side General Calibration Value Register (ADCx\_CLM0)

For more information, see CLMD register description.

Address: 4003\_B000h base + 6Ch offset = 4003\_B06Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																CLM0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

#### ADCx\_CLM0 field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLM0	Calibration Value Calibration Value

## 23.5 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.

## **23.5.1 Clock select and divide control**

One of four clock sources can be selected as the clock source for the ADC module.

This clock source is then divided by a configurable value to generate the input clock ADCK, to the module. The clock is selected from one of the following sources by means of CFG1[ADICLK].

- Bus clock. This is the default selection following reset.
- Bus clock divided by two. For higher bus clock rates, this allows a maximum divide-by-16 of the bus clock using CFG1[ADIV].
- ALTCLK: As defined for this MCU. See the chip configuration information. Conversions are possible using ALTCLK as the input clock source while the MCU is in Normal Stop mode.
- Asynchronous clock (ADACK): This clock is generated from a clock source within the ADC module. When the ADACK clock source is selected, it is not required to be active prior to conversion start. When it is selected and it is not active prior to a conversion start CFG2[ADACKEN]=0, ADACK is activated at the start of a conversion and deactivated when conversions are terminated. In this case, there is an associated clock startup delay each time the clock source is re-activated. To avoid the conversion time variability and latency associated with the ADACK clock startup, set CFG2[ADACKEN]=1 and wait the worst-case startup time of 5  $\mu$ 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.

### 23.5.2 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.

### 23.5.3 Hardware trigger and channel selects

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when  $SC2[ADTRG]$  is set and a hardware trigger select event, ADHWTSn, has occurred.

This source is not available on all MCUs. See the chip-specific ADC information for information on the ADHWT source and the ADHWTSn configurations specific to this MCU.

When an ADHWT source is available and hardware trigger is enabled, that is  $SC2[ADTRG]=1$ , a conversion is initiated on the rising-edge of ADHWT after a hardware trigger select event, that is, ADHWTSn, has occurred. If a conversion is in progress when a rising-edge of a trigger occurs, the rising-edge is ignored. In continuous convert configuration, only the initial rising-edge to launch continuous conversions is observed, and until conversion is aborted, the ADC continues to do conversions on the same SCn register that initiated the conversion. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

The hardware trigger select event, ADHWTSn, must be set prior to the receipt of the ADHWT signal. If these conditions are not met, the converter may ignore the trigger or use the incorrect configuration. If a hardware trigger select event is asserted during a conversion, it must stay asserted until the end of current conversion and remain set until the receipt of the ADHWT signal to trigger a new conversion. The channel and status fields selected for the conversion depend on the active trigger select signal:

- ADHWTS<sub>A</sub> active selects SC<sub>1A</sub>.
- ADHWTS<sub>n</sub> active selects SC<sub>1n</sub>.

### 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:

- ADHWTS<sub>A</sub> active selects RA register
- ADHWTS<sub>n</sub> active selects R<sub>n</sub> 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.

## 23.5.4 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

### 23.5.4.1 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, ADHWTS<sub>n</sub>, has occurred. The channel and status fields selected depend on the active trigger select signal:
  - ADHWTS<sub>A</sub> active selects SC1A.

- ADHWTSn active selects SC1n.
- if neither is active, the off condition is selected

### Note

Selecting more than one ADHWTSn prior to a conversion completion will result in unknown results. To avoid this, select only one ADHWTSn prior to a conversion completion.

- Following the transfer of the result to the data registers when continuous conversion is enabled, that is, when  $SC3[ADCO] = 1$ .

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, that is, when  $SC2[ADTRG] = 0$ , continuous conversions begin after SC1A is written and continue until aborted. In hardware triggered operation, that is, when  $SC2[ADTRG] = 1$  and one ADHWTSn event has occurred, continuous conversions begin after a hardware trigger event and continue until aborted.

If hardware averaging is enabled, a new conversion is automatically initiated after the completion of the current conversion until the correct number of conversions are completed. In software triggered operation, conversions begin after SC1A is written. In hardware triggered operation, conversions begin after a hardware trigger. If continuous conversions are also enabled, a new set of conversions to be averaged are initiated following the last of the selected number of conversions.

### 23.5.4.2 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.

### 23.5.4.3 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.

### 23.5.4.4 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  $f_{ADCK}$ .



### 23.5.4.5 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.

## Functional description

If the bus frequency is less than  $f_{ADCK}$ , precise sample time for continuous conversions cannot be guaranteed when short sample is enabled, that is, when  $CFG1[ADLSMP]=0$ .

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by  $CFG1[ADICLK]$ , and the divide ratio is specified by  $CFG1[ADIV]$ .

The maximum total conversion time for all configurations is summarized in the equation below. See the following tables for the variables referenced in the equation.

$$\text{ConversionTime} = \text{SFCAdder} + \text{AverageNum} \times (\text{BCT} + \text{LSTAdder} + \text{HSCAdder})$$

**Equation 1. Conversion time equation**

**Table 23-5. 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 23-6. 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 23-7. 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 23-7. Base conversion time (BCT) (continued)**

Mode	Base conversion time (BCT)
16b single-ended	25 ADCK cycles
16b differential	34 ADCK cycles

**Table 23-8. 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 23-9. 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.

## 23.5.4.6 Conversion time examples

The following examples use the [Equation 1 on page 362](#), and the information provided in [Table 23-5](#) through [Table 23-9](#).

### 23.5.4.6.1 Typical conversion time configuration

A typical configuration for ADC conversion is:

- 10-bit mode, with the bus clock selected as the input clock source
- The input clock divide-by-1 ratio selected
- Bus frequency of 8 MHz
- Long sample time disabled
- High-speed conversion disabled

The conversion time for a single conversion is calculated by using the [Equation 1 on page 362](#), and the information provided in [Table 23-5](#) through [Table 23-9](#). The table below lists the variables of [Equation 1 on page 362](#).

**Table 23-10. 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.

### 23.5.4.6.2 Long conversion time configuration

A configuration for long ADC conversion is:

- 16-bit differential mode with the bus clock selected as the input clock source
- The input clock divide-by-8 ratio selected
- Bus frequency of 8 MHz
- Long sample time enabled
- Configured for longest adder
- High-speed conversion disabled
- Average enabled for 32 conversions

The conversion time for this conversion is calculated by using the [Equation 1 on page 362](#), and the information provided in [Table 23-5](#) through [Table 23-9](#). The following table lists the variables of the [Equation 1 on page 362](#).

**Table 23-11. 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.

### 23.5.4.6.3 Short conversion time configuration

A configuration for short ADC conversion is:

- 8-bit Single-Ended mode with the bus clock selected as the input clock source
- The input clock divide-by-1 ratio selected
- Bus frequency of 20 MHz
- Long sample time disabled
- High-speed conversion enabled

The conversion time for this conversion is calculated by using the [Equation 1 on page 362](#), and the information provided in [Table 23-5](#) through [Table 23-9](#). The table below lists the variables of [Equation 1 on page 362](#).

**Table 23-12. 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.

### 23.5.4.7 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.

**23.5.5 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 23-13. 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.

## 23.5.6 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](#).

### 23.5.7 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.

## 23.5.8 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.

$$\text{Temp} = 25 - \left( (V_{\text{TEMP}} - V_{\text{TEMP25}}) \div m \right)$$

**Equation 2. 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).

### 23.5.9 MCU wait mode operation

Wait mode is a lower-power consumption Standby mode from which recovery is fast because the clock sources remain active.

If a conversion is in progress when the MCU enters Wait mode, it continues until completion. Conversions can be initiated while the MCU is in Wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two; and ADACK are available as conversion clock sources while in Wait mode. The use of ALTCLK as the conversion clock source in Wait is dependent on the definition of ALTCLK for this MCU. See the Chip Configuration information on ALTCLK specific to this MCU.

If the compare and hardware averaging functions are disabled, a conversion complete event sets  $SC1n[COCO]$  and generates an ADC interrupt to wake the MCU from Wait mode if the respective ADC interrupt is enabled, that is, when  $SC1n[AIEN]=1$ . If the hardware averaging function is enabled,  $SC1n[COCO]$  will set, and generate an interrupt if enabled, when the selected number of conversions are completed. If the compare function is enabled,  $SC1n[COCO]$  will set, and generate an interrupt if enabled, only if the compare conditions are met. If a single conversion is selected and the compare trigger is not met, the ADC will return to its idle state and cannot wake the MCU from Wait mode unless a new conversion is initiated by the hardware trigger.

### 23.5.10 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.

### 23.5.10.1 Normal Stop mode with ADACK disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its Idle state. The contents of the ADC registers, including Rn, are unaffected by Normal Stop mode. After exiting from Normal Stop mode, a software or hardware trigger is required to resume conversions.

### 23.5.10.2 Normal Stop mode with ADACK enabled

If ADACK is selected as the conversion clock, the ADC continues operation during Normal Stop mode. See the chip-specific ADC information for configuration information for this device.

If a conversion is in progress when the MCU enters Normal Stop mode, it continues until completion. Conversions can be initiated while the MCU is in Normal Stop mode by means of the hardware trigger or if continuous conversions are enabled.

If the compare and hardware averaging functions are disabled, a conversion complete event sets SC1n[COCO] and generates an ADC interrupt to wake the MCU from Normal Stop mode if the respective ADC interrupt is enabled, that is, when SC1n[AIEN]=1. The result register, Rn, will contain the data from the first completed conversion that occurred during Normal Stop mode. If the hardware averaging function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, when the selected number of conversions are completed. If the compare function is enabled, SC1n[COCO] will set, and generate an interrupt if enabled, only if the compare conditions are met. If a single conversion is selected and the compare is not true, the ADC will return to its idle state and cannot wake the MCU from Normal Stop mode unless a new conversion is initiated by another hardware trigger.

## 23.6 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 23-8](#), [Table 23-9](#), and [Table 23-10](#).

## Note

Hexadecimal values are designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

### 23.6.1 ADC module initialization example

#### 23.6.1.1 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.

#### 23.6.1.2 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.

## Initialization information

### 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 $V_{REFH}$ and $V_{REFL}$ ).

### 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.

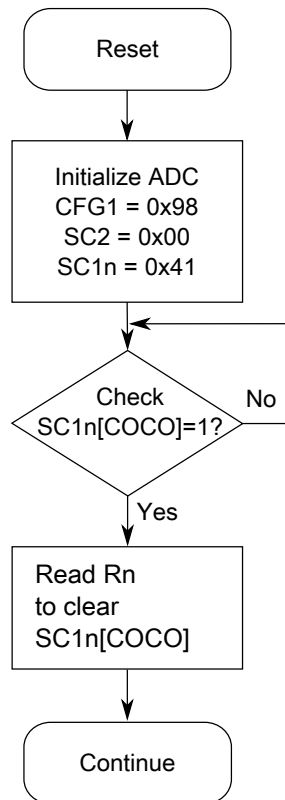


Figure 23-2. Initialization flowchart example

## 23.7 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](#).

### 23.7.1 External pins and routing

#### 23.7.1.1 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.

#### 23.7.1.2 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  $V_{ALTTL}$ . These voltage references are

selected using SC2[REFSEL]. The alternate voltage reference pair,  $V_{ALTH}$  and  $V_{ALTL}$ , may select additional external pins or internal sources based on MCU configuration. See the chip configuration information on the voltage references specific to this MCU.

In some packages, the external or alternate pairs are connected in the package to  $V_{DDA}$  and  $V_{SSA}$ , respectively. One of these positive references may be shared on the same pin as  $V_{DDA}$  on some devices. One of these ground references may be shared on the same pin as  $V_{SSA}$  on some devices.

If externally available, the positive reference may be connected to the same potential as  $V_{DDA}$  or may be driven by an external source to a level between the minimum Ref Voltage High and the  $V_{DDA}$  potential. The positive reference must never exceed  $V_{DDA}$ . If externally available, the ground reference must be connected to the same voltage potential as  $V_{SSA}$ . The voltage reference pairs must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the  $V_{REFH}$  and  $V_{REFL}$  loop. The best external component to meet this current demand is a 0.1  $\mu\text{F}$  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.

### 23.7.1.3 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.

## 23.7.2 Sources of error

### 23.7.2.1 Sampling error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy.

$$RAS + RADIN = SC / (FMAX * NUMTAU * CADIN)$$

**Figure 23-3. Sampling equation**

Where:

RAS = External analog source resistance

SC = Number of ADCK cycles used during sample window

CADIN = Internal ADC input capacitance

NUMTAU =  $-\ln(\text{LSBERR} / 2^N)$

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.

### 23.7.2.2 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.

### 23.7.2.3 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  $V_{\text{REFH}}$  to  $V_{\text{REFL}}$ .
- There is a 0.1  $\mu\text{F}$  low-ESR capacitor from  $V_{\text{DDA}}$  to  $V_{\text{SSA}}$ .
- If inductive isolation is used from the primary supply, an additional 1  $\mu\text{F}$  capacitor is placed from  $V_{\text{DDA}}$  to  $V_{\text{SSA}}$ .
- $V_{\text{SSA}}$ , and  $V_{\text{REFL}}$ , if connected, is connected to  $V_{\text{SS}}$  at a quiet point in the ground plane.
- Operate the MCU in Wait or Normal Stop mode before initiating (hardware-triggered conversions) or immediately after initiating (hardware- or software-triggered conversions) the ADC conversion.
  - For software triggered conversions, immediately follow the write to SC1 with a Wait instruction or Stop instruction.
  - For Normal Stop mode operation, select ADACK as the clock source. Operation in Normal Stop reduces  $V_{\text{DD}}$  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  $V_{\text{DD}}$  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 ( $C_{\text{AS}}$ ) on the selected input channel to  $V_{\text{REFL}}$  or  $V_{\text{SSA}}$ . 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.

### 23.7.2.4 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:

$$1\text{LSB} = (V_{\text{REFH}}) / 2^N$$

**Equation 3. Ideal code width for an N-bit converter**

There is an inherent quantization error due to the digitization of the result. For 8-bit, 10-bit, or 12-bit conversions, the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be  $\pm 1/2$  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.

### 23.7.2.5 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.

### 23.7.2.6 Code jitter, non-monotonicity, and missing codes

Analog-to-digital converters are susceptible to three special forms of error:

- **Code jitter:** Code jitter occurs when a given input voltage converts to one of the two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code, and vice-versa. However, even small amounts of system noise can cause the converter to be indeterminate, between two codes, for a range of input voltages around the transition voltage.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally, the techniques discussed in [Noise-induced errors](#) reduces this error.

- **Non-monotonicity:** Non-monotonicity occurs when, except for code jitter, the converter converts to a lower code for a higher input voltage.
- **Missing codes:** Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

# Chapter 24

## Comparator (CMP)

### 24.1 Chip-specific CMP information

#### 24.1.1 CMP instantiation information

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

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

The CMP can be optionally on in all modes except VLLS0.

The CMP has several module-to-module interconnects in order to facilitate ADC triggering, TPM triggering, and infrared interfaces. For complete details on the CMP module interconnects, see the [Module-to-Module section](#).

The CMP does not support window compare function and a 0 must always be written to CMP\_CR1[WE]. The sample function has limited functionality since the SAMPLE input to the block is not connected to a valid input. Usage of sample operation is limited to a divided version of the bus clock (CMP\_CR1[SE] = 0).

Due to the pin number limitation, the CMP pass through mode is not supported by this device, so the CMPx\_MUXCR[PSTM] must be left as 0.

## 24.1.2 CMP input connections

The following table shows the fixed internal connections to the CMP0.

**Table 24-1. CMP input connections**

CMP inputs	CMP0
IN0	CMP0_IN0
IN1	CMP0_IN1
IN2	CMP0_IN2
IN3	CMP0_IN3
IN4	12-bit DAC0 reference/ CMP0_IN4
IN5	CMP0_IN5
IN6	Bandgap <sup>1</sup>
IN7	6-bit DAC0 reference

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

## 24.1.3 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. When using VREF\_OUT, any ADC conversion using this same reference at the same time is negatively impacted.
- VDD– $V_{in2}$  input

## 24.1.4 CMP trigger mode

The CMP and 6-bit DAC sub-block supports trigger mode operation when CMP\_CR1[TRIGM] is set. When trigger mode is enabled, the trigger event will initiate a compare sequence that must first enable the CMP and DAC prior to performing a CMP operation and capturing the output. In this device, control for this two-staged sequencing is provided from the LPTMR. The LPTMR triggering output is always enabled when the LPTMR is enabled. The first signal is supplied to enable the CMP and DAC and is asserted at the same time as the TCF flag is set. The delay to the second signal that triggers the CMP to capture the result of the compare operation is dependent on the LPTMR configuration.

- In Time Counter mode with prescaler enabled, the delay is 1/2 Prescaler output period.
- In Time Counter mode with prescaler bypassed, the delay is 1/2 Prescaler clock period.

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

## 24.2 Introduction

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.

### 24.2.1 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
- Digitally filtered:
  - Filter can be bypassed
  - Can be clocked via scaled bus clock
- External hysteresis can be used at the same time that the output filter is used for internal functions
- Two software selectable performance levels:
  - Shorter propagation delay at the expense of higher power
  - Low power, with longer propagation delay
- DMA transfer support
  - A comparison event can be selected to trigger a DMA transfer
- Functional in all modes of operation
- The filter functions are not available in the following modes:
  - Stop
  - VLPS
  - LLS
  - VLLSx

### **24.2.2 6-bit DAC key features**

The 6-bit DAC has the following features:

- 6-bit resolution
- Selectable supply reference source
- Power Down mode to conserve power when not in use
- Option to route the output to internal comparator input

### **24.2.3 ANMUX key features**

The ANMUX has the following features:

- Two 8-to-1 channel mux
- Operational over the entire supply range



## 24.2.4 CMP, DAC and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.

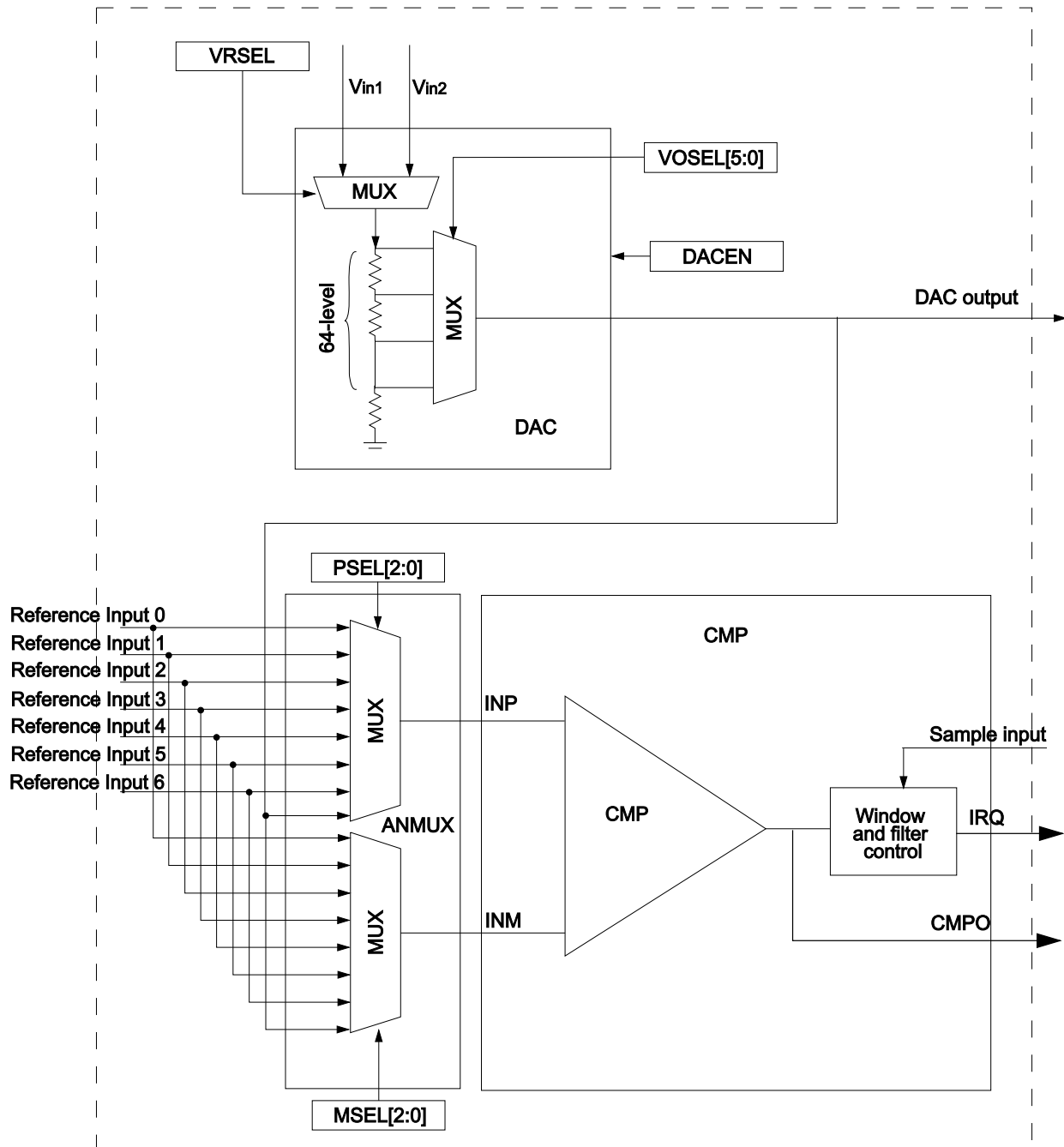
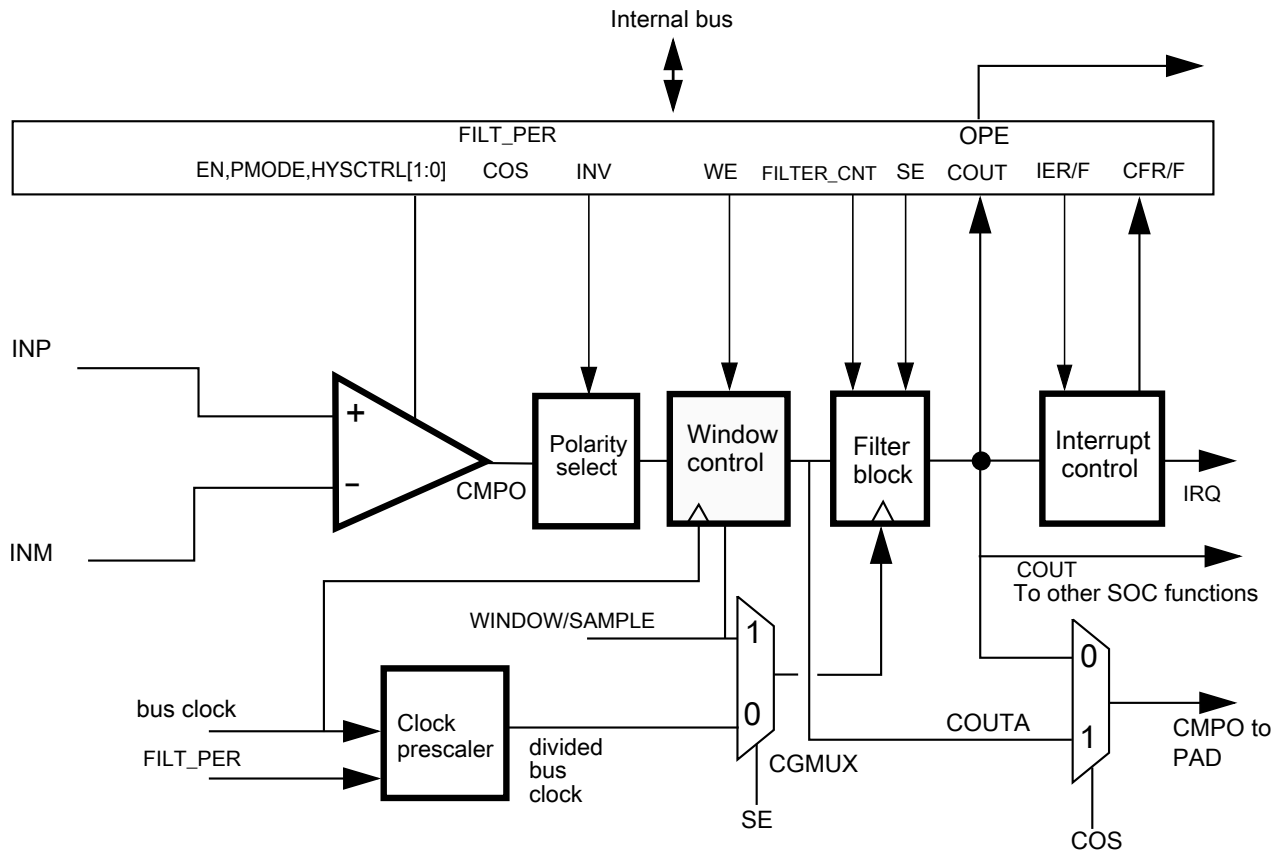


Figure 24-1. CMP, DAC and ANMUX block diagram

## 24.2.5 CMP block diagram

The following figure shows the block diagram for the CMP module.



**Figure 24-2. Comparator module block diagram**

In the CMP block diagram:

- The Window Control block is bypassed when  $CR1[WE] = 0$
- 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  $0x01$ .
  - $CR1[SE] = 0$ , the divided bus clock is used as sampling clock
- If enabled, the Filter block will incur up to one bus clock additional latency penalty on COUT due to the fact that COUT, which is crossing clock domain boundaries, must be resynchronized to the bus clock.

## 24.3 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="#">24.3.1/387</a>
4007_3001	CMP Control Register 1 (CMP0_CR1)	8	R/W	00h	<a href="#">24.3.2/388</a>
4007_3002	CMP Filter Period Register (CMP0_FPR)	8	R/W	00h	<a href="#">24.3.3/389</a>
4007_3003	CMP Status and Control Register (CMP0_SCR)	8	R/W	00h	<a href="#">24.3.4/390</a>
4007_3004	DAC Control Register (CMP0_DACCR)	8	R/W	00h	<a href="#">24.3.5/391</a>
4007_3005	MUX Control Register (CMP0_MUXCR)	8	R/W	00h	<a href="#">24.3.6/391</a>

### 24.3.1 CMP Control Register 0 (CMPx\_CR0)

Address: 4007\_3000h base + 0h offset = 4007\_3000h

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 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. 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.

Table continues on the next page...

**CMPx\_CR0 field descriptions (continued)**

Field	Description
HYSTCTR	<p>Comparator hard block hysteresis control</p> <p>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.</p> <p>00 Level 0 01 Level 1 10 Level 2 11 Level 3</p>

**24.3.2 CMP Control Register 1 (CMPx\_CR1)**

Address: 4007\_3000h base + 1h offset = 4007\_3001h

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	<p>Sample Enable</p> <p>SE must be clear to 0 and usage of sample operation is limited to a divided version of the bus clock.</p> <p>0 Sampling mode is not selected. 1 Sampling mode is selected.</p>
6 WE	<p>Windowing Enable</p> <p>The CMP does not support window compare function and a 0 must always be written to WE.</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>

*Table continues on the next page...*

**CMPx\_CR1 field descriptions (continued)**

Field	Description
	See the electrical specifications table in the device Data Sheet for details. 0 Low-Speed (LS) Comparison mode selected. In this mode, CMP has slower output propagation delay and lower current consumption. 1 High-Speed (HS) Comparison mode selected. In this mode, CMP has faster output propagation delay and higher current consumption.
3 INV	Comparator INVERT  Allows selection of the polarity of the analog comparator function. It is also driven to the COUT output, on both the device pin and as SCR[COUT], when OPE=0.  0 Does not invert the comparator output. 1 Inverts the comparator output.
2 COS	Comparator Output Select  0 Set the filtered comparator output (CMPO) to equal COUT. 1 Set the unfiltered comparator output (CMPO) to equal COUTA.
1 OPE	Comparator Output Pin Enable  0 CMPO is not available on the associated CMPO output pin. If the comparator does not own the pin, this field has no effect. 1 CMPO is available on the associated CMPO output pin.  The comparator output (CMPO) is driven out on the associated CMPO output pin if the comparator owns the pin. If the comparator does not own the field, this bit has no effect.
0 EN	Comparator Module Enable  Enables the Analog Comparator module. When the module is not enabled, it remains in the off state, and consumes no power. When the user selects the same input from analog mux to the positive and negative port, the comparator is disabled automatically.  0 Analog Comparator is disabled. 1 Analog Comparator is enabled.

**24.3.3 CMP Filter Period Register (CMPx\_FPR)**

Address: 4007\_3000h base + 2h offset = 4007\_3002h

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
FILT_PER	Filter Sample Period  Specifies the sampling period, in bus clock cycles, of the comparator output filter, when CR1[SE]=0. Setting FILT_PER to 0x0 disables the filter. Filter programming and latency details appear in the <a href="#">Functional description</a> .

### 24.3.4 CMP Status and Control Register (CMPx\_SCR)

Address: 4007\_3000h base + 3h offset = 4007\_3003h

Bit	7	6	5	4	3	2	1	0
Read	0	DMAEN	0	IER	IEF	CFR	CFF	COUT
Write						w1c	w1c	
Reset	0	0	0	0	0	0	0	0

**CMPx\_SCR field descriptions**

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 DMAEN	DMA Enable Control  Enables the DMA transfer triggered from the CMP module. When this field is set, a DMA request is asserted when CFR or CFF is set.  0 DMA is disabled. 1 DMA is enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 IER	Comparator Interrupt Enable Rising  Enables the CFR interrupt from the CMP. When this field is set, an interrupt will be asserted when CFR is set.  0 Interrupt is disabled. 1 Interrupt is enabled.
3 IEF	Comparator Interrupt Enable Falling  Enables the CFF interrupt from the CMP. When this field is set, an interrupt will be asserted when CFF is set.  0 Interrupt is disabled. 1 Interrupt is enabled.
2 CFR	Analog Comparator Flag Rising  Detects a rising-edge on COUT, when set, during normal operation. CFR is cleared by writing 1 to it. During Stop modes, CFR is edge sensitive .  0 Rising-edge on COUT has not been detected. 1 Rising-edge on COUT has occurred.
1 CFF	Analog Comparator Flag Falling  Detects a falling-edge on COUT, when set, during normal operation. CFF is cleared by writing 1 to it. During Stop modes, CFF is edge sensitive .  0 Falling-edge on COUT has not been detected. 1 Falling-edge on COUT has occurred.

Table continues on the next page...

**CMPx\_SCR field descriptions (continued)**

Field	Description
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.

**24.3.5 DAC Control Register (CMPx\_DACCR)**

Address: 4007\_3000h base + 4h offset = 4007\_3004h

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.  0 DAC is disabled. 1 DAC is enabled.
6 VRSEL	Supply Voltage Reference Source Select  0 $V_{in1}$ is selected as resistor ladder network supply reference. 1 $V_{in2}$ is selected as resistor ladder network supply reference.
VOSEL	DAC Output Voltage Select  Selects an output voltage from one of 64 distinct levels.  $DACO = (V_{in} / 64) * (VOSEL[5:0] + 1)$ , so the DACO range is from $V_{in} / 64$ to $V_{in}$ .

**24.3.6 MUX Control Register (CMPx\_MUXCR)**

Address: 4007\_3000h base + 5h offset = 4007\_3005h

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

**CMPx\_MUXCR field descriptions**

Field	Description
7 PSTM	Pass Through Mode Enable

*Table continues on the next page...*

**CMPx\_MUXCR field descriptions (continued)**

Field	Description
	<p>This bit is used to enable to MUX pass through mode. Pass through mode is always available but for some devices this feature must be always disabled due to the lack of package pins.</p> <p>0 Pass Through Mode is disabled. 1 Pass Through Mode is enabled.</p>
6 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
5–3 PSEL	<p>Plus Input Mux Control</p> <p>Determines which input is selected for the plus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams.</p> <p><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.</p> <p>000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7</p>
MSEL	<p>Minus Input Mux Control</p> <p>Determines which input is selected for the minus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams.</p> <p><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.</p> <p>000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7</p>

## 24.4 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[COOUT].

### 24.4.1 CMP functional modes

There are the following main sub-blocks to the CMP module:

- The comparator itself
- The filter function

The filter, CR0[FILTER\_CNT], can be clocked from an internal clock source only. 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 comparator filter and sampling features can be combined as shown in the following table. Individual modes are discussed below.

**Table 24-2. 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	
3B	1	0	0	0x01	> 0x00	<b>Sampled, Non-Filtered mode</b> See the <a href="#">Sampled, Non-Filtered mode (#s 3B)</a> .
4B	1	0	0	> 0x01	> 0x00	<b>Sampled, Filtered mode</b> See the <a href="#">Sampled, Filtered mode (#s 4B)</a> .
All other combinations of CR1[EN], CR1[WE], CR1[SE], CR0[FILTER_CNT], and FPR[FILT_PER] are illegal.						

For cases where a comparator is used to drive a fault input, for example, for a 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.

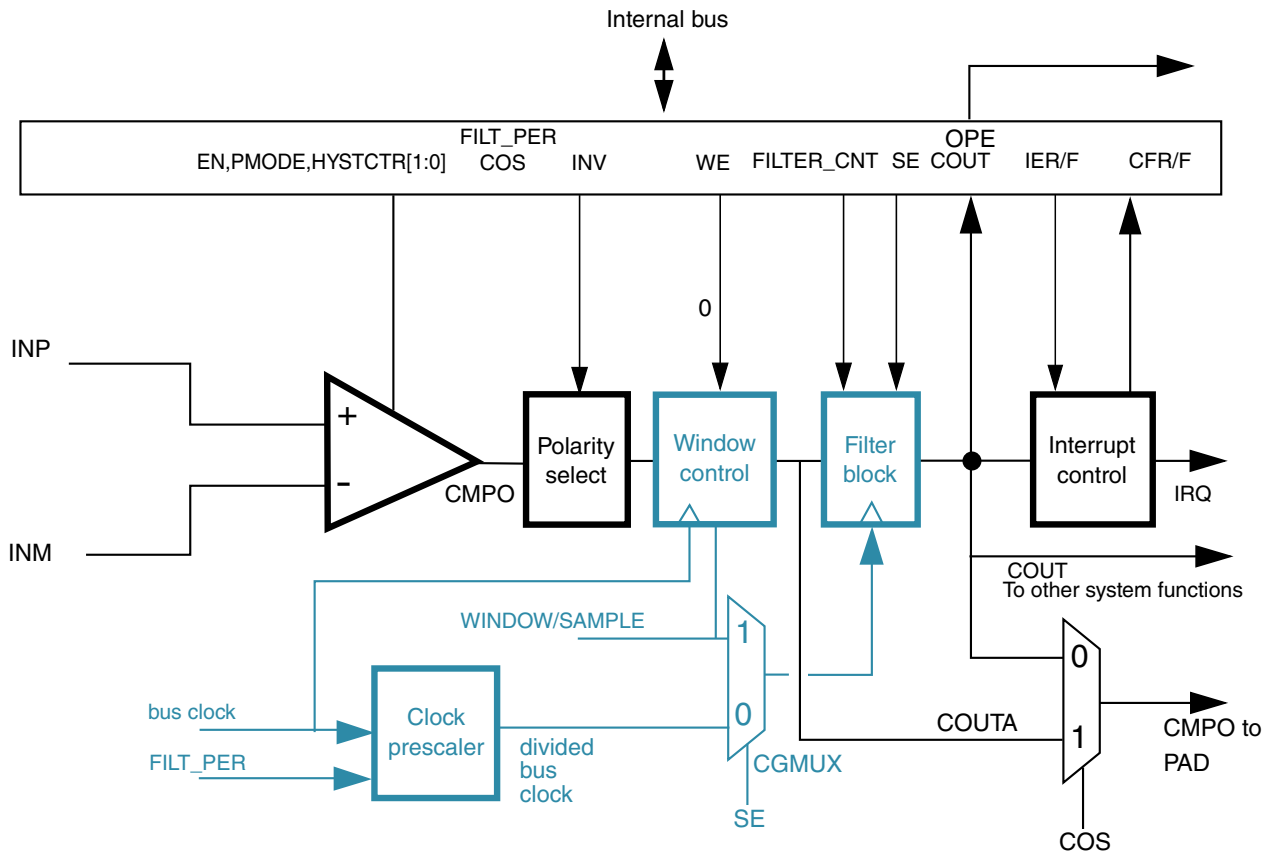
**Note**

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.

**24.4.1.1 Disabled mode (# 1)**

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

**24.4.1.2 Continuous mode (#s 2A & 2B)**



**Figure 24-3. Comparator operation in Continuous mode**

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.

### 24.4.1.3 Sampled, Non-Filtered mode (#s 3B)

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.

The comparator filter has no other function than sample/hold of the comparator output in this mode (# 3B).

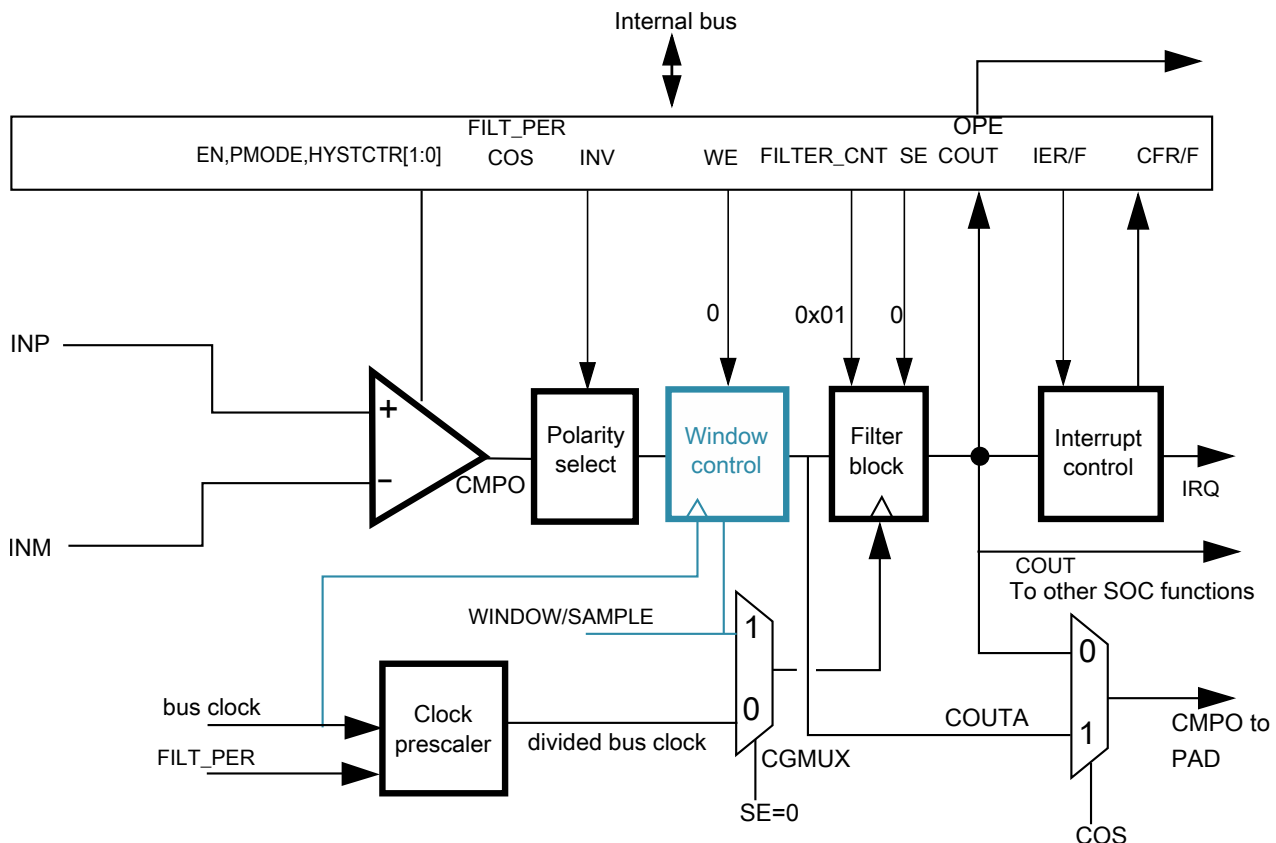
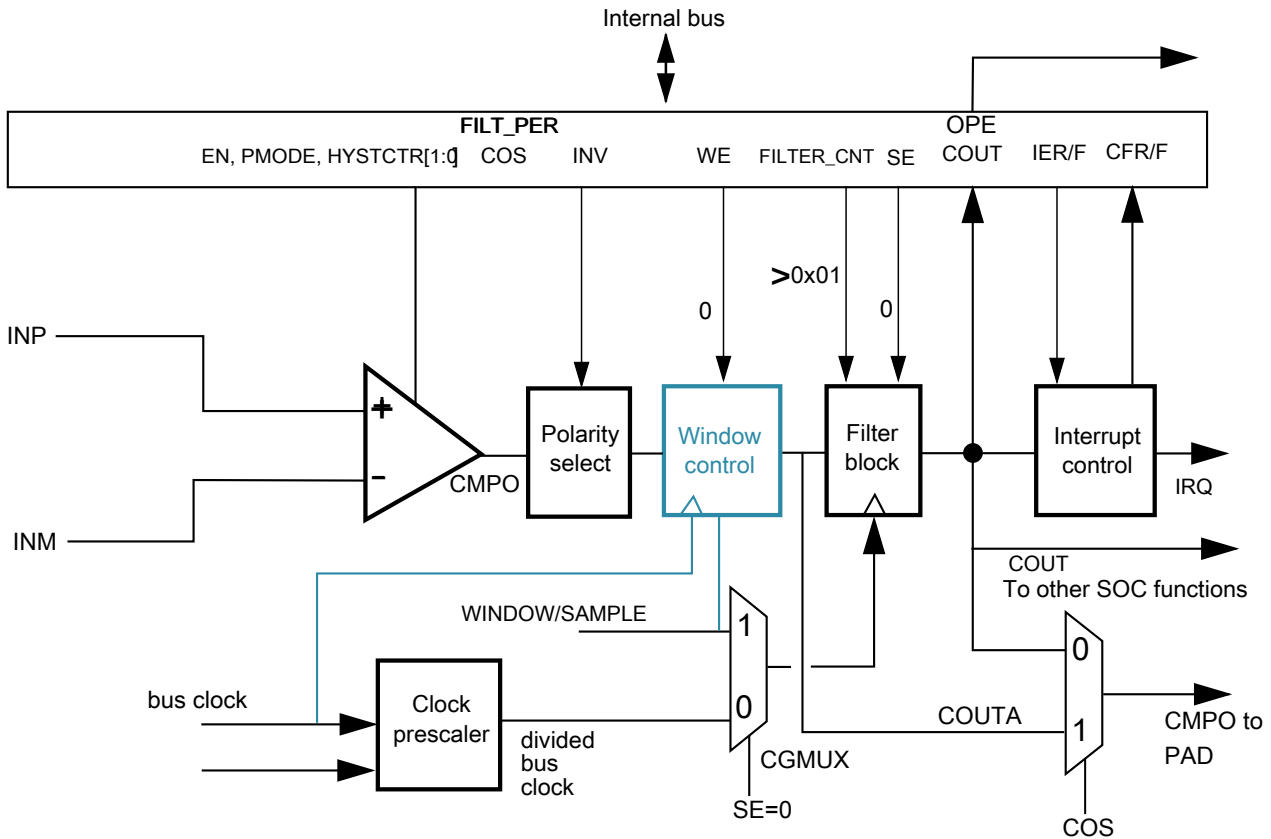


Figure 24-4. Sampled, Non-Filtered (# 3B): sampling interval internally derived

### 24.4.1.4 Sampled, Filtered mode (#s 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.



**Figure 24-5. 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.

## 24.4.2 Power modes

### 24.4.2.1 Wait mode operation

During Wait and VLPW modes, the CMP, if enabled, continues to operate normally and a CMP interrupt can wake the MCU.

### 24.4.2.2 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.

### 24.4.2.3 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].

### 24.4.2.4 Background Debug Mode Operation

When the microcontroller is in active background debug mode, the CMP continues to operate normally.

## 24.4.3 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 and filter. See the Data Sheets for power-on delays of the comparators. 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.

#### **24.4.4 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 modes. Filtering can be performed using an internal timebase defined by FPR[FILT\_PER] 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.

##### **24.4.4.1 Enabling filter modes**

Filter modes can be enabled by:

- Setting CR0[FILTER\_CNT] > 0x01 and
- Setting FPR[FILT\_PER] to a nonzero value

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 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.

#### 24.4.4.2 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 24-3. 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 <sub>PD</sub>
2B	1	0	0	X	0x00		
3B	1	0	0	0x01	> 0x00	Sampled, Non-Filtered mode	T <sub>PD</sub> + (FPR[FILT_PER] * T <sub>per</sub> ) + T <sub>per</sub>
4B	1	0	0	> 0x01	> 0x00	Sampled, Filtered mode	T <sub>PD</sub> + (CR0[FILTER_CNT] * FPR[FILT_PER] x T <sub>per</sub> ) + T <sub>per</sub>

1. T<sub>PD</sub> represents the intrinsic delay of the analog component plus the polarity select logic. T<sub>per</sub> is the period of the bus clock.

## 24.5 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

## 24.6 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.

## 24.7 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.

## 24.8 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.

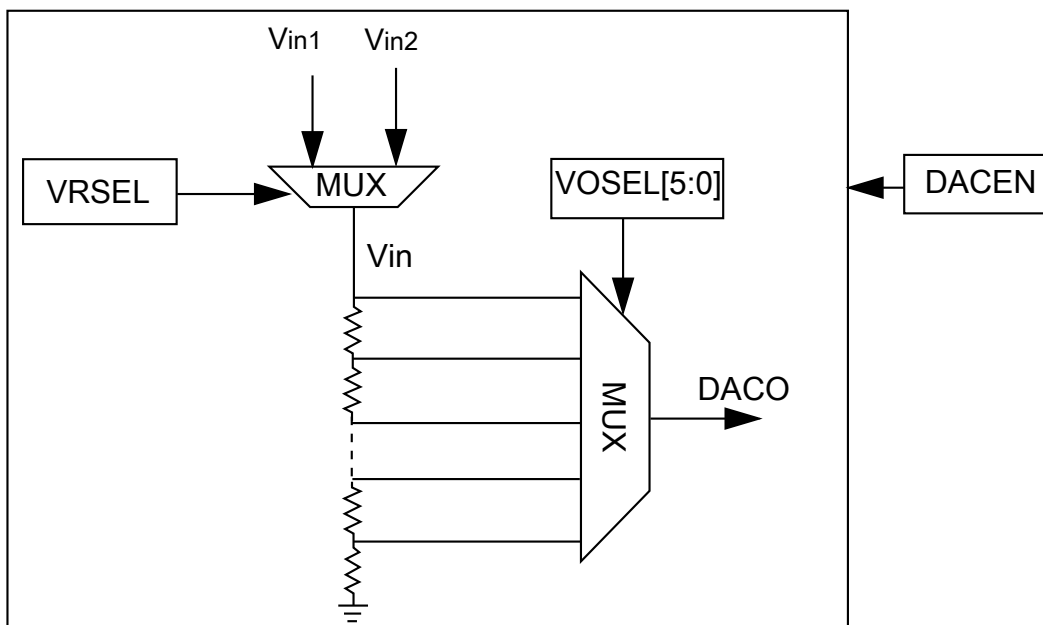


Figure 24-6. 6-bit DAC block diagram

## 24.9 DAC functional description

This section provides DAC functional description information.

## 24.9.1 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

## 24.10 DAC resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

## 24.11 DAC clocks

This module has a single clock input, the bus clock.

## 24.12 DAC interrupts

This module has no interrupts.

## 24.13 CMP Trigger Mode

CMP and DAC are configured to CMP Trigger mode when `CMP_CR1[TRIGM]` is set to 1.

In addition, the CMP must be enabled. If the DAC is to be used as a reference to the CMP, it must also be enabled.

CMP Trigger mode depends on an external timer resource to periodically enable the CMP and 6-bit DAC in order to generate a triggered compare.

Upon setting `TRIGM`, the CMP and DAC are placed in a standby state until an external timer resource trigger is received.

# Chapter 25

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

### 25.1 Introduction

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.

### 25.2 Features

The features of the DAC module include:

- On-chip programmable reference generator output. The voltage output range is from  $1/4096 V_{in}$  to  $V_{in}$ , and the step is  $1/4096 V_{in}$ , where  $V_{in}$  is the input voltage.
- $V_{in}$  can be selected from two reference sources
- Static operation in Normal Stop mode
- 2-word data buffer supported with multiple operation modes
- DMA support

### 25.3 Block diagram

The block diagram of the DAC module is as follows:

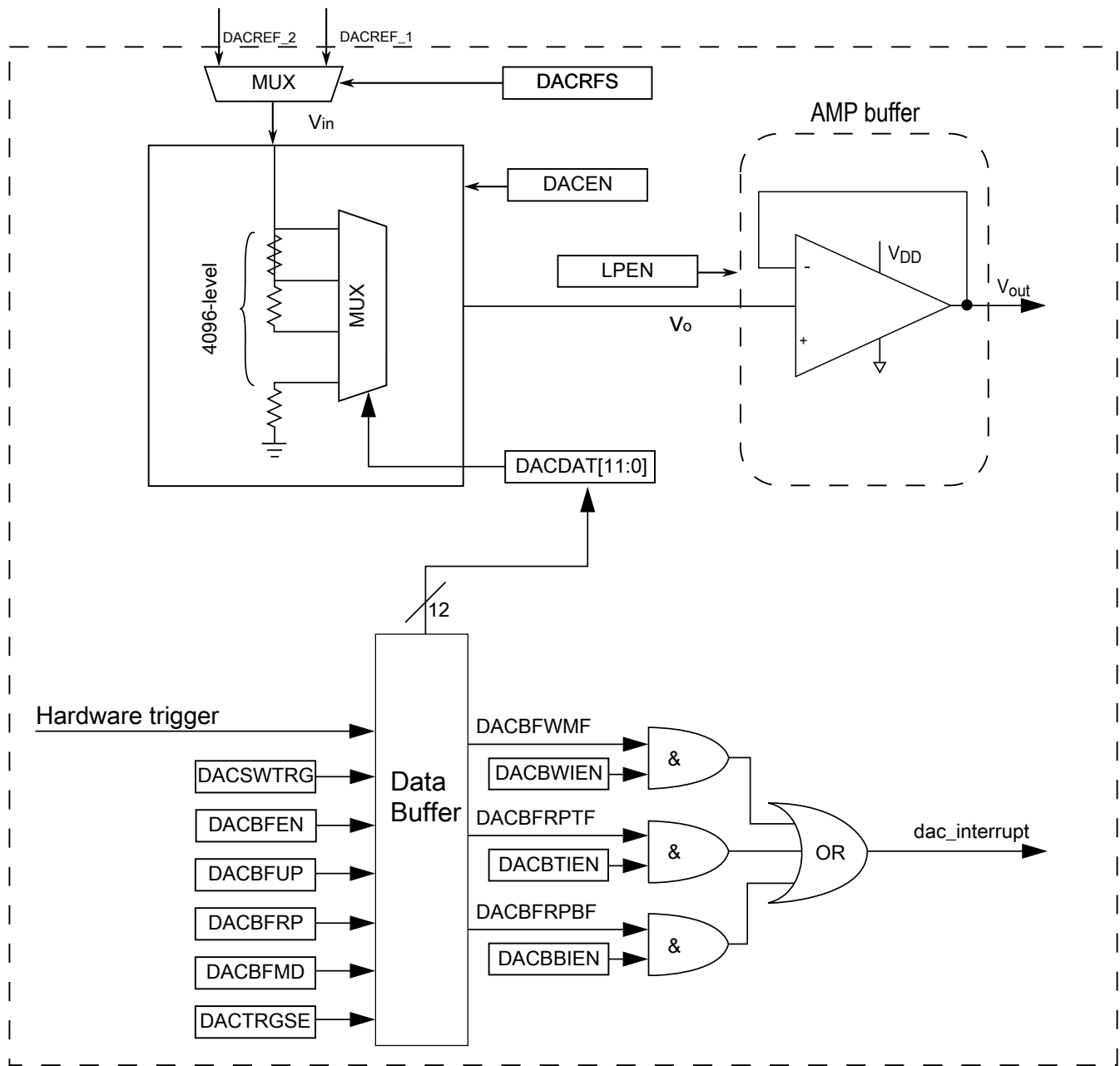


Figure 25-1. DAC block diagram

## 25.4 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="#">25.4.1/405</a>
4003_F001	DAC Data High Register (DAC0_DAT0H)	8	R/W	00h	<a href="#">25.4.2/405</a>
4003_F002	DAC Data Low Register (DAC0_DAT1L)	8	R/W	00h	<a href="#">25.4.1/405</a>
4003_F003	DAC Data High Register (DAC0_DAT1H)	8	R/W	00h	<a href="#">25.4.2/405</a>
4003_F020	DAC Status Register (DAC0_SR)	8	R/W	<a href="#">See section</a>	<a href="#">25.4.3/406</a>
4003_F021	DAC Control Register (DAC0_C0)	8	R/W	00h	<a href="#">25.4.4/407</a>
4003_F022	DAC Control Register 1 (DAC0_C1)	8	R/W	00h	<a href="#">25.4.5/408</a>
4003_F023	DAC Control Register 2 (DAC0_C2)	8	R/W	01h	<a href="#">25.4.6/408</a>

## 25.4.1 DAC Data Low Register (DACx\_DATnL)

Address: 4003\_F000h base + 0h offset + (2d × i), where i=0d to 1d

Bit	7	6	5	4	3	2	1	0
Read	DATA0							
Write	DATA0							
Reset	0	0	0	0	0	0	0	0

## DACx\_DATnL field descriptions

Field	Description
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>

## 25.4.2 DAC Data High Register (DACx\_DATnH)

Address: 4003\_F000h base + 1h offset + (2d × i), where i=0d to 1d

Bit	7	6	5	4	3	2	1	0
Read	0				DATA1			
Write	0				DATA1			
Reset	0	0	0	0	0	0	0	0

## 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>
DATA1	DATA1

Table continues on the next page...

**DACx\_DATnH field descriptions (continued)**

Field	Description
	When the DAC Buffer is not enabled, DATA[11:0] controls the output voltage based on the following formula. $V_{out} = V_{in} * (1 + DACDAT0[11:0])/4096$
	When the DAC buffer is enabled, DATA[11:0] is mapped to the 16-word buffer.

**25.4.3 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					0	DACBFRPT	DACBFRPB
Write							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 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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].

## 25.4.4 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	0	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 DACSWTRG	<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 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</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> <p>0 The DAC buffer read pointer bottom flag interrupt is disabled. 1 The DAC buffer read pointer bottom flag interrupt is enabled.</p>

### 25.4.5 DAC Control Register 1 (DACx\_C1)

Address: 4003\_F000h base + 22h offset = 4003\_F022h

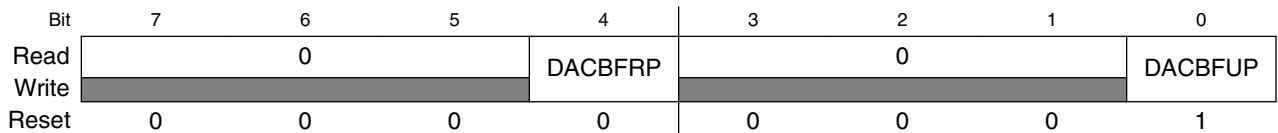


#### DACx\_C1 field descriptions

Field	Description
7 DMAEN	DMA Enable Select 0 DMA is disabled. 1 DMA is enabled. When DMA is enabled, the DMA request will be generated by original interrupts. The interrupts will not be presented on this module at the same time.
6-3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2-1 DACBFMD	DAC Buffer Work Mode Select 00 Normal mode 01 Reserved 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.

### 25.4.6 DAC Control Register 2 (DACx\_C2)

Address: 4003\_F000h base + 23h offset = 4003\_F023h



#### DACx\_C2 field descriptions

Field	Description
7-5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...



**DACx\_C2 field descriptions (continued)**

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

## 25.5 Functional description

The 12-bit DAC module can select one of the two reference inputs—DACREF\_1 and DACREF\_2 as the DAC reference voltage,  $V_{in}$  by C0 [DACRFS]. See the chip-specific DAC information to determine the source options for DACREF\_1 and DACREF\_2.

When the DAC is enabled, it converts the data in DACDAT0[11:0] or the data from the DAC data buffer to a stepped analog output voltage. The output voltage range is from  $V_{in}$  to  $V_{in}/4096$ , and the step is  $V_{in}/4096$ .

### 25.5.1 DAC data buffer operation

When the DAC is enabled and the buffer is not enabled, the DAC module always converts the data in DAT0 to the analog output voltage.

When both the DAC and the buffer are enabled, the DAC converts the data in the data buffer to analog output voltage. The data buffer read pointer advances to the next word whenever a hardware or software trigger event occurs.

The data buffer can be configured to operate in Normal mode, One-Time Scan mode 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].

### 25.5.1.1 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.

### 25.5.1.2 Modes of DAC data buffer operation

The following table describes the different modes of data buffer operation for the DAC module.

**Table 25-1. Modes of DAC data buffer operation**

Modes	Description
Buffer Normal mode	This is the default mode. The buffer works as a circular buffer. The read pointer increases by one, every time the trigger occurs. When the read pointer reaches the upper limit, it goes to 0 directly in the next trigger event.
Buffer One-time Scan mode	The read pointer increases by 1 every time the trigger occurs. When it reaches the upper limit, it stops there. If read pointer is reset to the address other than the upper limit, it will increase to the upper address and stop there again. <b>NOTE:</b> If the software set the read pointer to the upper limit, the read pointer will not advance in this mode.
FIFO Mode	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. <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. <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 & 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

**Table 25-1. Modes of DAC data buffer operation**

Modes	Description
	<p>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>

## 25.5.2 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.

## 25.5.3 Resets

During reset, the DAC is configured in the default mode and is disabled.

## 25.5.4 Low-Power mode operation

The following table shows the wait mode and the stop mode operation of the DAC module.

**Table 25-2. 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 26

## Voltage Reference (VREFV1)

### 26.1 Introduction

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.

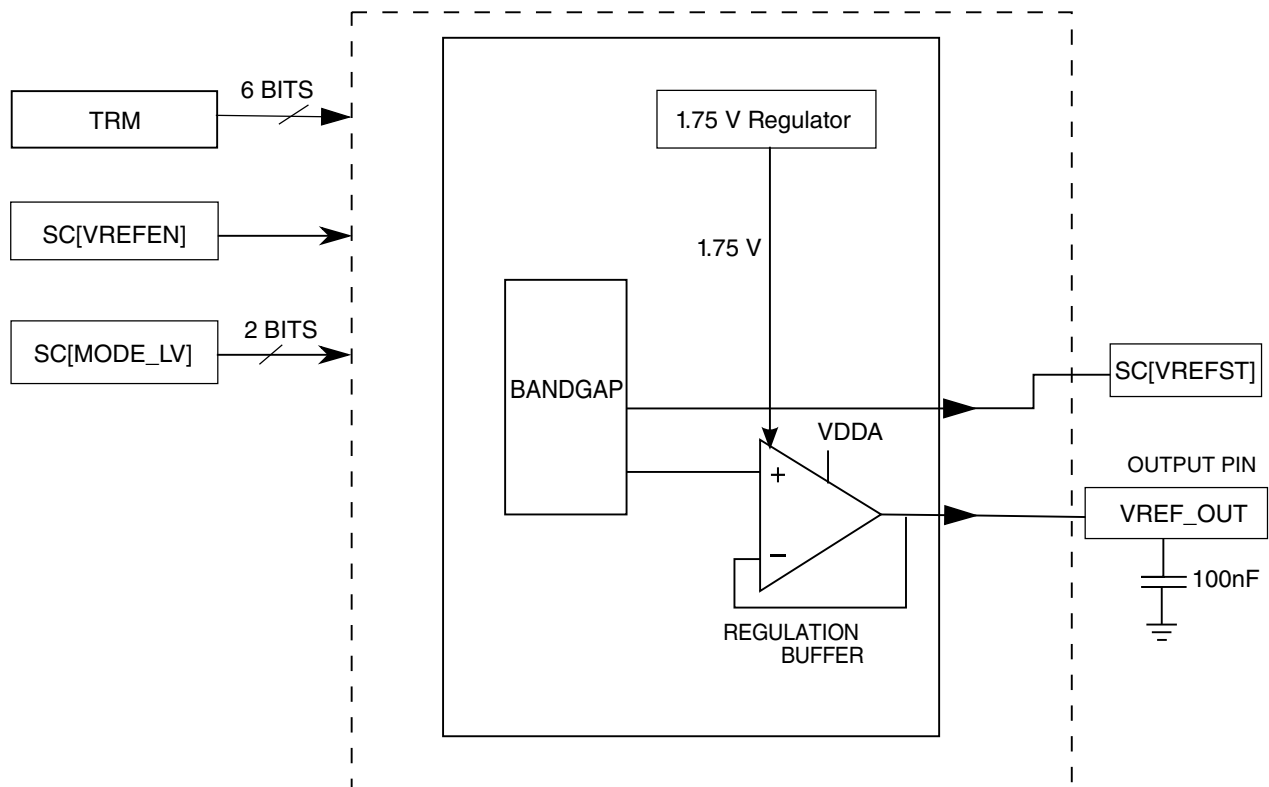


Figure 26-1. Voltage reference block diagram

## 26.1.1 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 signal is output 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.

## 26.1.2 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
- VREF\_OUT output signal

## 26.1.3 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.

## 26.1.4 VREF Signal Descriptions

The following table shows the Voltage Reference signals properties.

**Table 26-1. VREF Signal Descriptions**

Signal	Description	I/O
VREF_OUT	Internally-generated Voltage Reference output	O

### NOTE

The VREF\_OUT signal does not have a dedicated pin; it is combined with VREFH. When the VREF output buffer is disabled, the status of the VREF\_OUT signal is high-impedence.

## 26.2 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	<a href="#">See section</a>	<a href="#">26.2.1/416</a>
4007_4001	VREF Status and Control Register (VREF_SC)	8	R/W	00h	<a href="#">26.2.2/417</a>

## 26.2.1 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 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.
TRIM	Trim bits  These bits change the resulting VREF by approximately ± 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.  000000 Min .... .... 111111 Max



## 26.2.2 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>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> <p>0 Disabled 1 Enabled</p>
4 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
3 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
2 VREFST	<p>Internal Voltage Reference stable</p> <p>This bit indicates that the bandgap reference within the Voltage Reference module has completed its startup and stabilization.</p> <p><b>NOTE:</b> This bit is valid only when the chop oscillator is not being used.</p>

Table continues on the next page...

## VREF\_SC field descriptions (continued)

Field	Description
	0 The module is disabled or not stable. 1 The module is stable.
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

## 26.3 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 26-2. 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

### 26.3.1 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.

## 26.3.2 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.

### 26.3.2.1 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, Tchop\_osc\_stup.

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 (Tstup) and the value is specified in the appropriate device data sheet.

### 26.3.2.2 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 (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.

### 26.3.2.3 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.

### 26.3.2.4 SC[MODE\_LV] = 11

Reserved

## 26.4 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.

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

## 26.5 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 ( $T_{\text{chop\_osc\_stup}}$ ) 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 ( $T_{\text{stup}}$ ) 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.



# Chapter 27

## Multipurpose Clock Generator Lite (MCG\_Lite)

### 27.1 Introduction

The Multipurpose Clock Generator Lite (MCG\_Lite) module provides several clock source options for the MCU. This module contains one 48 MHz and one 8/2 MHz Internal Reference Clock (IRC) sources. The module selects one of IRCs or External Oscillator/Clock (EXT) as the MCU clock sources.

#### 27.1.1 Features

The MCG\_Lite module has the following features:

- High-frequency Internal Reference Clock (HIRC)
  - 48 MHz clock source
  - Support various trims to achieve target accuracy
- Low-frequency Internal Reference Clock (LIRC)
  - Selectable 8 MHz or 2 MHz clock source
  - Trim bit to ensure  $\pm 3\%$  accuracy across PVT
- Glitchless clock switcher for internal clock sources (HIRC and LIRC) and external clock source (EXT)
- Dedicated high frequency clock sources output (MCGPCLK) for peripheral use
- Dedicated low frequency clock sources output (LIRC\_CLK) for peripheral use
- Divider/prescaler (FCRDIV) for low frequency IRC to support /1, /2, /4, /8, /16, /32, /64, and /128 division factors
- Second divider (LIRC\_DIV2) to support /1, /2, /4, /8, /16, /32, /64, and /128 division factors
- Control signal for the external clock/oscillator (EXT)
  - EREFS0, HGO0, RANGE0
  - External clock from crystal oscillator can be used as system clock sources

## 27.1.2 Block diagram

The block diagram of MCG\_Lite is as follows.

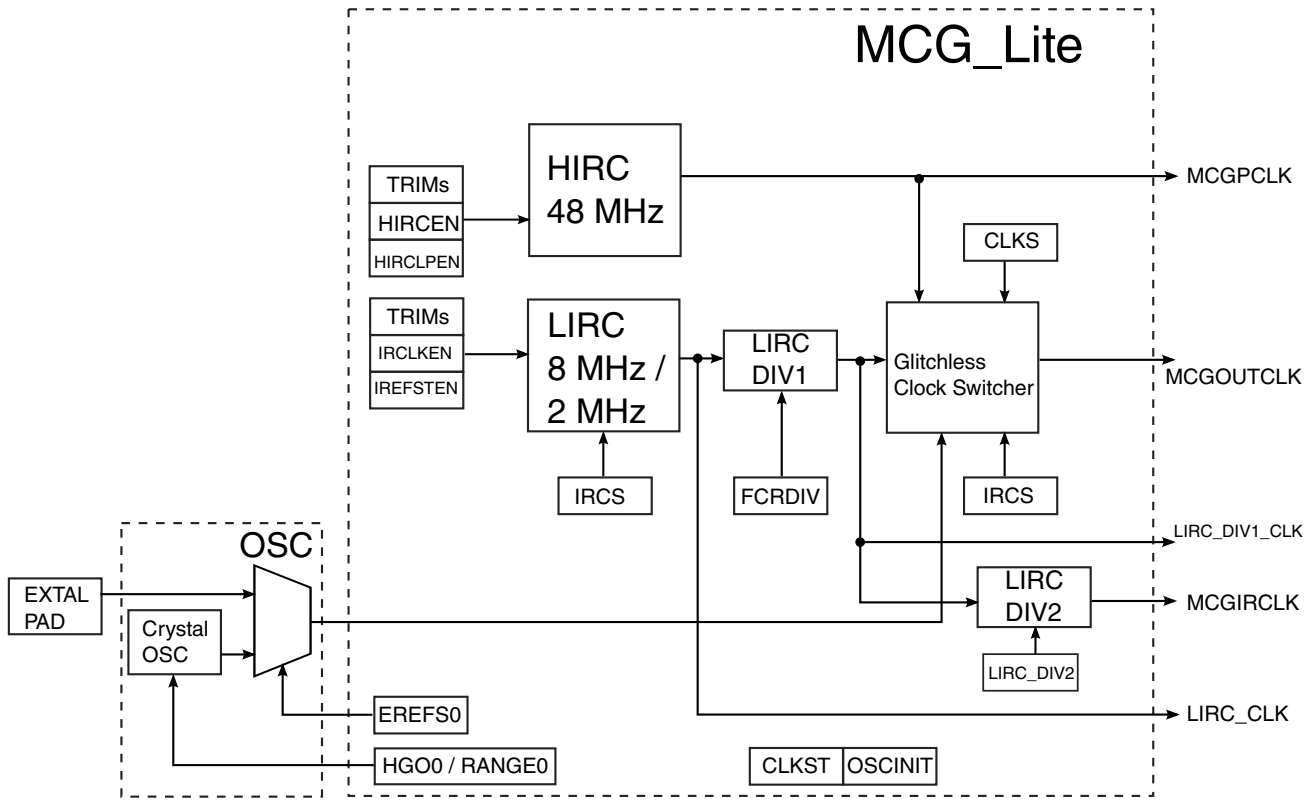


Figure 27-1. MCG\_Lite block diagram

## 27.2 Memory map and register definition

The MCG\_Lite module contains several fields for selecting the clock source and the dividers for various module clocks.

### NOTE

The MCG\_Lite registers can be written only in supervisor mode. Write accesses in user mode are blocked and will result in a bus error.

### MCG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4006_4000	MCG Control Register 1 (MCG_C1)	8	R/W	40h	<a href="#">27.2.1/425</a>

Table continues on the next page...

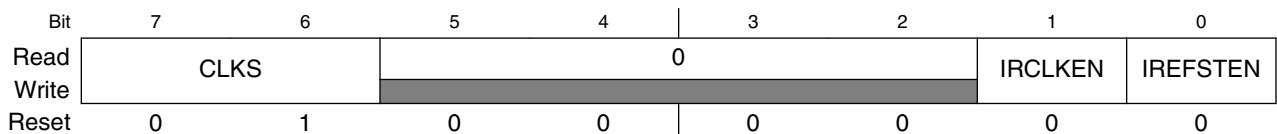


## MCG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4006_4001	MCG Control Register 2 (MCG_C2)	8	R/W	01h	<a href="#">27.2.2/426</a>
4006_4006	MCG Status Register (MCG_S)	8	R	04h	<a href="#">27.2.3/427</a>
4006_4008	MCG Status and Control Register (MCG_SC)	8	R/W	00h	<a href="#">27.2.4/427</a>
4006_4018	MCG Miscellaneous Control Register (MCG_MC)	8	R/W	00h	<a href="#">27.2.5/428</a>

## 27.2.1 MCG Control Register 1 (MCG\_C1)

Address: 4006\_4000h base + 0h offset = 4006\_4000h



## MCG\_C1 field descriptions

Field	Description
7–6 CLKS	<p>Clock Source Select</p> <p>Selects the clock source for MCGOUTCLK.</p> <p>00 Selects HIRC clock as the main clock source. This is HIRC mode.            01 Selects LIRC clock as the main clock source. This is LIRC2M or LIRC8M mode.            10 Selects external clock as the main clock source. This is EXT mode.            11 Reserved. Writing 11 takes no effect.</p>
5–2 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
1 IRCLKEN	<p>Internal Reference Clock Enable</p> <p>Enables the IRC source.</p> <p>0 LIRC is disabled.            1 LIRC is enabled.</p>
0 IREFSTEN	<p>Internal Reference Stop Enable</p> <p>Controls whether the IRC source remains enabled when the MCG_Lite enters Stop mode.</p> <p>0 LIRC is disabled in Stop mode.            1 LIRC is enabled in Stop mode, if IRCLKEN is set.</p>

## 27.2.2 MCG Control Register 2 (MCG\_C2)

Address: 4006\_4000h base + 1h offset = 4006\_4001h

Bit	7	6	5	4	3	2	1	0
Read	0		RANGE0		HGO0	EREFS0	0	
Write								
Reset	0	0	0	0	0	0	0	1

### MCG\_C2 field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 RANGE0	External Clock Source Frequency Range Select  Selects the frequency for the crystal oscillator or the external clock source. See the Oscillator (OSC) chapter for more details and refer to the chip datasheet for the frequency ranges used.  00 Low frequency range selected for the crystal oscillator or the external clock source. 01 High frequency range selected for the crystal oscillator or the external clock source. 10 Very high frequency range selected for the crystal oscillator or the external clock source. 11 Very high frequency range selected for the crystal oscillator or the external clock source. Same effect as 10.
3 HGO0	Crystal Oscillator Operation Mode Select  Selects the crystal oscillator mode of operation. See the Oscillator (OSC) chapter for more details.  0 Configure crystal oscillator for low-power operation. 1 Configure crystal oscillator for high-gain operation.
2 EREFS0	External Clock Source Select  Selects the source for the external reference clock. See the Oscillator (OSC) chapter for more details.  0 External clock requested. 1 Oscillator requested.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 IRCS	Low-frequency Internal Reference Clock Select  Controls the LIRC to work at 2 MHz or 8 MHz mode.  0 LIRC is in 2 MHz mode. 1 LIRC is in 8 MHz mode.

## 27.2.3 MCG Status Register (MCG\_S)

Address: 4006\_4000h base + 6h offset = 4006\_4006h

Bit	7	6	5	4	3	2	1	0
Read	0				CLKST		OSCINIT0	0
Write								
Reset	0	0	0	0	0	1	0	0

### MCG\_S field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–2 CLKST	Clock Mode Status  Indicates the current clock mode. This field does not update immediately after a write to MCG_C1[CLKS] due to internal synchronization between clock domains.  00 HIRC clock is selected as the main clock source, and MCG_Lite works at HIRC mode. 01 LIRC clock is selected as the main clock source, and MCG_Lite works at LIRC2M or LIRC8M mode. 10 External clock is selected as the main clock source, and MCG_Lite works at EXT mode. 11 Reserved.
1 OSCINIT0	OSC Initialization Status  This flag, which resets to 0, is set to 1 after the initialization cycles of the crystal oscillator clock are completed. After being set, the bit is cleared to 0 if the OSC is subsequently disabled. See the Oscillator (OSC) chapter for more information.  0 OSC is not ready. 1 OSC clock is ready.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

## 27.2.4 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	0				FCRDIV			0
Write								
Reset	0	0	0	0	0	0	0	0

### MCG\_SC field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

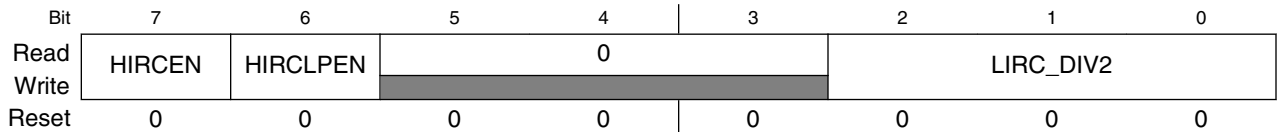
*Table continues on the next page...*

**MCG\_SC field descriptions (continued)**

Field	Description
3-1 FCRDIV	<p>Low-frequency Internal Reference Clock Divider</p> <p>Selects the factor value to divide the LIRC source.</p> <p>000 Division factor is 1.                      001 Division factor is 2.                      010 Division factor is 4.                      011 Division factor is 8.                      100 Division factor is 16.                      101 Division factor is 32.                      110 Division factor is 64.                      111 Division factor is 128.</p>
0 Reserved	<p>This field is reserved.                      This read-only field is reserved and always has the value 0.</p>

**27.2.5 MCG Miscellaneous Control Register (MCG\_MC)**

Address: 4006\_4000h base + 18h offset = 4006\_4018h



**MCG\_MC field descriptions**

Field	Description
7 HIRCEN	<p>High-frequency IRC Enable</p> <p>Enables the HIRC, even when MCG_Lite is not working at HIRC mode.</p> <p>0 HIRC source is not enabled.                      1 HIRC source is enabled.</p>
6 HIRCLPEN	<p>High-frequency IRC Low-power Mode Enable</p> <p>0 HIRC is disabled in Low-power mode, such as Stop/VLPR/VLPW/VLPS mode.                      1 HIRC is enabled in Low-power mode, such as Stop/VLPR/VLPW/VLPS mode.</p>
5-3 Reserved	<p>This field is reserved.                      This read-only field is reserved and always has the value 0.</p>
LIRC_DIV2	<p>Second Low-frequency Internal Reference Clock Divider</p> <p>Selects the factor value to further divide the LIRC source.</p> <p>000 Division factor is 1.                      001 Division factor is 2.                      010 Division factor is 4.                      011 Division factor is 8.</p>

*Table continues on the next page...*

## MCG\_MC field descriptions (continued)

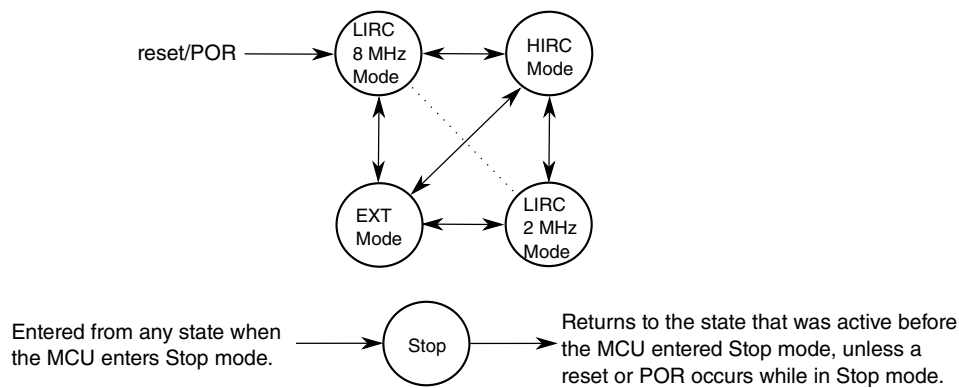
Field	Description
100	Division factor is 16.
101	Division factor is 32.
110	Division factor is 64.
111	Division factor is 128.

## 27.3 Functional description

This section presents the functional details of the MCG\_Lite module.

### 27.3.1 Clock mode switching

Different states of the MCG\_Lite module are shown in the following figure. The arrows indicate the permitted MCG\_Lite mode transitions.



**Figure 27-2. MCG\_Lite mode state diagram**

The MCG\_Lite module does not support switch between LIRC 2 MHz and 8 MHz directly, because 2 MHz and 8 MHz clock generators share circuits and logics. To switch between each other, the module must be in HIRC or EXT clock mode.

If entering VLPR mode with MCG\_MC[HIRCCLPEN] cleared, MCG\_Lite has to be configured and enter LIRC2M, LIRC8M or EXT mode, and MCG\_MC[HIRCEN] must also be cleared. 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.

When power on or out of reset, LIRC is selected as the main clock source. To select other clock sources, the user must perform the following steps.

To enter HIRC mode:

## Functional description

1. Write 1b to MCG\_MC[HIRCEN] to enable HIRC (optional).
2. Write 00b to MCG\_C1[CLKS] to select HIRC clock source.
3. Check MCG\_S[CLKST] to confirm HIRC clock source is selected.

To enter EXT mode:

1. Configure MCG\_C2[EREFS0] for external clock source selection.
2. Write 10b to MCG\_C1[CLKS] to select external clock source.
3. Check MCG\_S[CLKST] to confirm external clock source is selected.

To enter LIRC2M mode from HIRC or EXT mode:

1. Write 0b to MCG\_C2[IRCS] to select LIRC 2M.
2. Write 1b to MCG\_C1[IRCLKEN] to enable LIRC clock (optional).
3. Write 01b to MCG\_C1[CLKS] to select LIRC clock source.
4. Check MCG\_S[CLKST] to confirm LIRC clock source is selected.

To enter LIRC8M mode from HIRC or EXT mode:

1. Write 1b to MCG\_C2[IRCS] to select LIRC 8M.
2. Write 1b to MCG\_C1[IRCLKEN] to enable LIRC clock (optional).
3. Write 01b to MCG\_C1[CLKS] to select LIRC clock source.
4. Check MCG\_S[CLKST] to confirm LIRC clock source is selected.

### 27.3.2 LIRC divider 1

In the MCG\_Lite module, there is a divider for LIRC clock. The divider supports /1, /2, /4, /8, /16, /32, /64, and /128 division factors. For details, see the register field description of [MCG\\_SC\[FCRDIV\]](#). The divided clock of LIRC DIV1 is one of the inputs of clock select switch. It is the input for the 2nd LIRC DIV as well. See the Chip Configuration information for more details.

### 27.3.3 LIRC divider 2

In the MCG\_Lite module, there is another divider to further divide the LIRC clock, named LIRC DIV2. This divider supports /1, /2, /4, /8, /16, /32, /64, and /128 division factors. For details, see the register field description of [MCG\\_MC\[LIRC\\_DIV2\]](#). The divided clock of LIRC DIV2 is MCGIRCLK, and it can be used as peripheral clock. See the Chip Configuration information for more details.

### 27.3.4 Enable LIRC in Stop mode

In Stop mode, HIRC is disabled to save power. For LIRC, by default it is disabled as well. To enable LIRC in Stop mode, write 1b to MCG\_C1[IREFSTEN] and MCG\_C1[IRCLKEN] before entering Stop mode.

### 27.3.5 MCG-Lite in Low-power mode

In Stop/VLPS mode, MCG-Lite is inactive, HIRC is disabled except that MCG\_MC[HIRCEN] and MCG\_MC[HIRCLPEN] are set before entering this Stop/VLPS mode, and LIRC is disabled except that both MCG\_C1[IREFSTEN] and MCG\_C1[IRCLKEN] are set before entering the Stop/VLPS mode.

In LLS/VLLS mode, MCG-Lite is power down.

In VLPR/VLPW mode, MCG-Lite is in Low-power mode, HIRC is disabled except that MCG\_MC[HIRCEN] and MCG\_MC[HIRCLPEN] are set before entering this VLPR/VLPW mode. As internal voltage regulator is required in the HIRC logic, BGEN, BGBE and VLPO bits in the PMC\_REGSC register should also be set before entering VLPR, while LIRC can keep working.





# Chapter 28

## Oscillator (OSC)

### 28.1 Chip-specific OSC information

#### 28.1.1 OSC modes of operation with MCG\_Lite and RTC

The most common method of controlling the OSC block is through MCG\_C1[CLKS] and the fields of MCG\_C2 register to configure for crystal or external clock operation. OSC\_CR also provides control for enabling the OSC module and configuring internal load capacitors for the EXTAL and XTAL pins. See the [OSC](#) and [MCG\\_Lite](#) chapters for more details.

RTC\_CR[OSCE] has overriding control over the MCG\_Lite and OSC\_CR enable functions. When RTC\_CR[OSCE] is set, the OSC is configured for low frequency, low power and RTC\_CR[SCxP] override OSC\_CR[SCxP] to control the internal capacitance configuration. See the RTC chapter for more details.

### 28.2 Introduction

The OSC module is a crystal oscillator. The module, in conjunction with an external crystal or resonator, generates a reference clock for the MCU.

### 28.3 Features and Modes

Key features of the module are listed here.

- Supports 32 kHz crystals (Low Range mode)
- Supports 1–8 MHz,, 8–32 MHz crystals and resonators (High Range mode)

## Block Diagram

- Automatic Gain Control (AGC) to optimize power consumption in high frequency ranges 1–8 MHz,, 8–32 MHz using low-power mode
- High gain option in frequency ranges: 32 kHz, 1–8 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.

## 28.4 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 . The OSCCLK can only work in run mode. OSCERCLK and 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 28-1. OSC Module Block Diagram**

## 28.5 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 28-1. OSC Signal Descriptions**

Signal	Description	I/O
EXTAL	External clock/Oscillator input	I
XTAL	Oscillator output	O

## 28.6 External Crystal / Resonator Connections

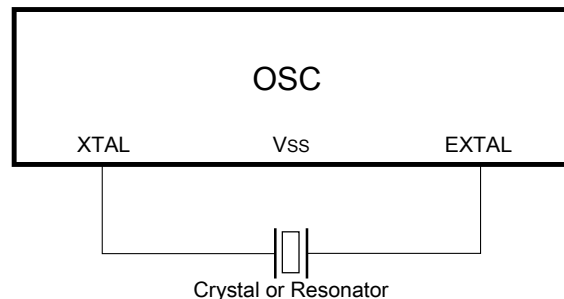
The connections for a crystal/resonator frequency reference are shown in the figures found here.

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. In addition, a series resistor ( $R_S$ ) may be used in high-gain modes. The following table shows all possible connections.

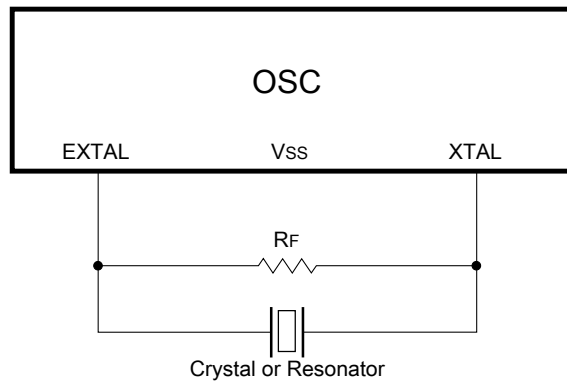
**Table 28-2. External Crystal/Resonator Connections**

Oscillator Mode	Connections
Low-frequency (32 kHz), low-power	Connection 1 <sup>1</sup>
Low-frequency (32 kHz), high-gain	Connection 2/Connection 4 <sup>2</sup> / Connection 3 <sup>3</sup>
High-frequency (1~32 MHz), low-power	Connection 3 <sup>1</sup>
High-frequency (1~32 MHz), high-gain	Connection 3/Connection 4 <sup>2</sup>

1. With the low-power mode, the oscillator has the internal feedback resistor  $R_F$ . Therefore, the feedback resistor must not be externally with the Connection 3.
2. When the frequency of the crystal is less than 2 MHz and the load capacitors ( $C_x$ ,  $C_y$ ) are less than 16 pF, use Connection 4.
3. When the load capacitors ( $C_x$ ,  $C_y$ ) are greater than 30 pF, use Connection 3.



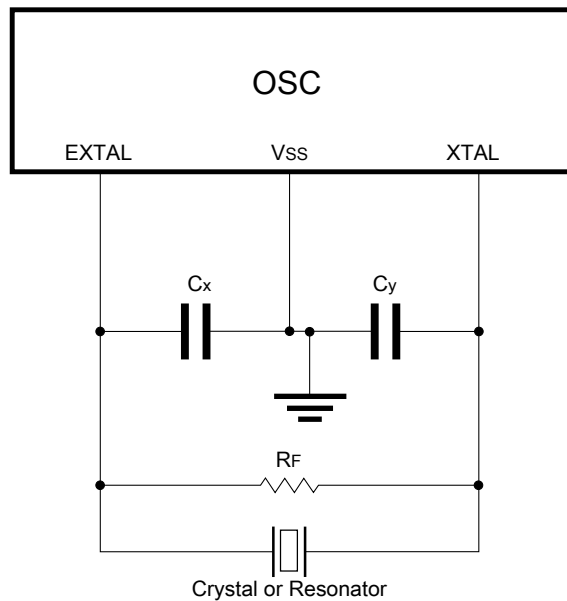
**Figure 28-2. Crystal/Ceramic Resonator Connections - Connection 1**



**Figure 28-3. 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 28-4. Crystal/Ceramic Resonator Connections - Connection 3**

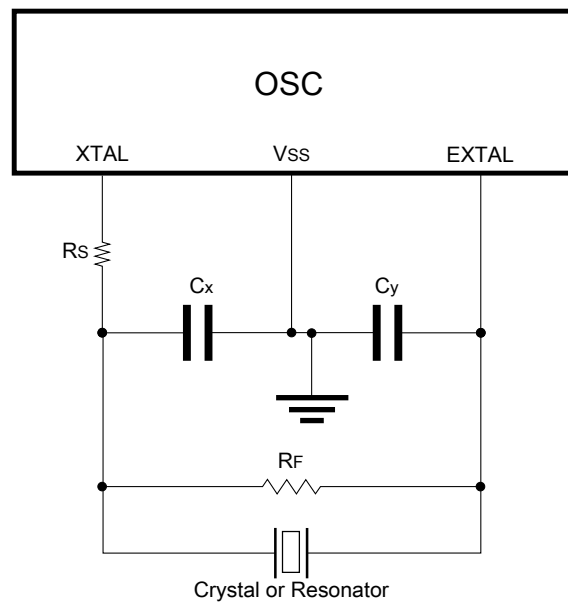


Figure 28-5. Crystal/Ceramic Resonator Connections - Connection 4

## 28.7 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.

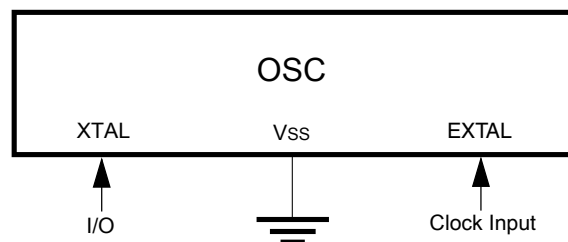


Figure 28-6. External Clock Connections

## 28.8 Memory Map/Register Definitions

Some oscillator module register bits are typically incorporated into other peripherals such as MCG or SIM.

## 28.8.1 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 (OSC0_CR)	8	R/W	00h	<a href="#">28.8.1.1/438</a>

### 28.8.1.1 OSC Control Register (OSCx\_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

#### OSCx\_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

Table continues on the next page...

**OSCx\_CR field descriptions (continued)**

<b>Field</b>	<b>Description</b>
	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.

## 28.9 Functional Description

Functional details of the module can be found here.

### 28.9.1 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.

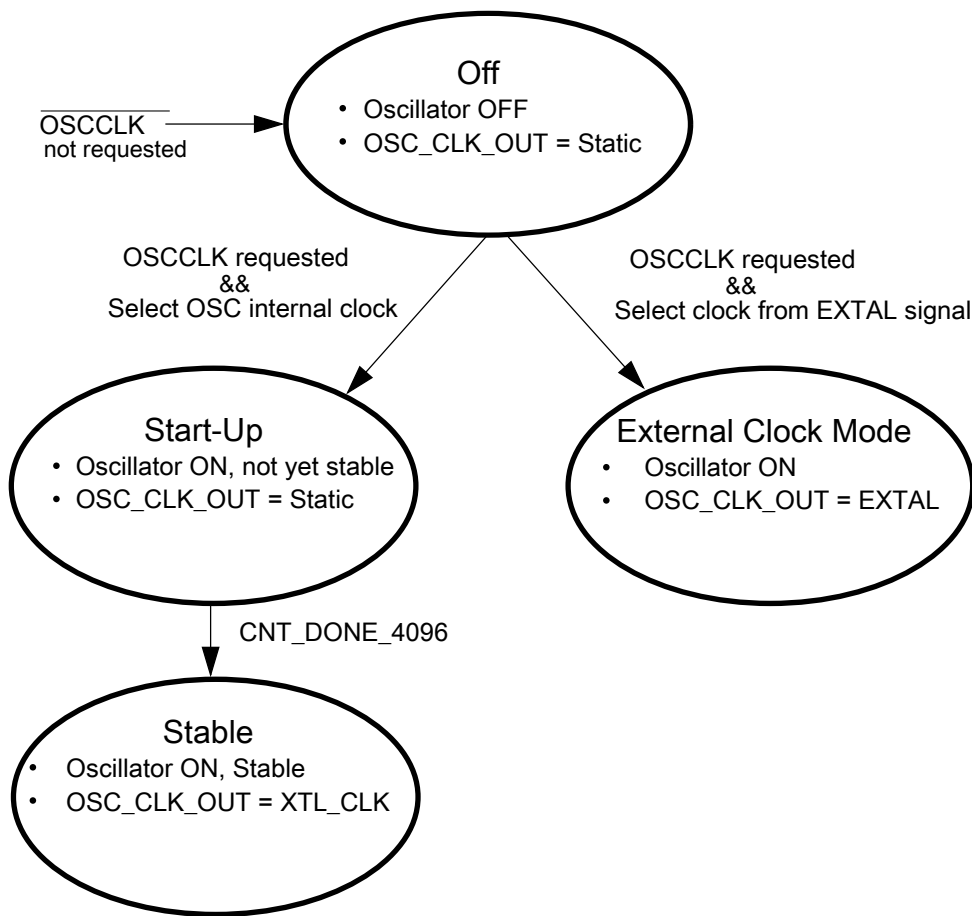


Figure 28-7. OSC Module state diagram

**NOTE**

XTL\_CLK is the clock generated internally from OSC circuits.

**28.9.1.1 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.



### 28.9.1.2 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.

### 28.9.1.3 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.

### 28.9.1.4 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.

## 28.9.2 OSC module modes

The OSC is a pierce-type oscillator that supports external crystals or resonators operating over the frequency ranges shown in [Table 28-3](#). These modes assume the following conditions: OSC is enabled to generate clocks (OSC\_EN=1), configured to generate clocks internally (MCG\_C2[EREFS] = 1), and some or one of the other peripherals (MCG, Timer, and so on) is configured to use the oscillator output clock (OSC\_CLK\_OUT).

**Table 28-3. 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 28-3. Oscillator modes (continued)**

Mode	Frequency Range
High-frequency mode1, high-gain	$f_{osc\_hi\_1}$ (1 MHz) up to $f_{osc\_hi\_1}$ (8 MHz)
High-frequency mode1, low-power	
High-frequency mode2, high-gain	$f_{osc\_hi\_2}$ (8 MHz) up to $f_{osc\_hi\_2}$ (32 MHz)
High-frequency mode2, low-power	

**NOTE**

For information about low power modes of operation used in this chip and their alignment with some OSC modes, see the chip's Power Management details.

**28.9.2.1 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.

**28.9.2.2 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.

### 28.9.2.3 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.

### 28.9.2.4 High-Frequency, Low-Power Mode

In high-frequency, low-power mode, the oscillator uses a gain control loop to minimize power consumption. As the oscillation amplitude increases, the amplifier current is reduced. This continues until a desired amplitude is achieved at steady-state. In this mode, no external resistor should be used.

The oscillator input buffer in this mode is differential. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels.

## 28.9.3 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.

## 28.9.4 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.

## 28.10 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.

## 28.11 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.

## 28.12 Interrupts

The OSC module does not generate any interrupts.

# Chapter 29

## Timer/PWM Module (TPM)

### 29.1 Chip-specific TPM information

The following registers are not available in this device:

**Table 29-1. TPM register**

Absolute address	Register	Instance
4003_901C	Channel (n) Status and Control (TPM1_C2SC)	TPM1
4003_9020	Channel (n) Value (TPM1_C2V)	TPM1
4003_9024	Channel (n) Status and Control (TPM1_C3SC)	TPM1
4003_9028	Channel (n) Value (TPM1_C3V)	TPM1
4003_902C	Channel (n) Status and Control (TPM1_C4SC)	TPM1
4003_9030	Channel (n) Value (TPM1_C4V)	TPM1
4003_9034	Channel (n) Status and Control (TPM1_C5SC)	TPM1
4003_9038	Channel (n) Value (TPM1_C5V)	TPM1
4003_A01C	Channel (n) Status and Control (TPM2_C2SC)	TPM2
4003_A020	Channel (n) Value (TPM2_C2V)	TPM2
4003_A024	Channel (n) Status and Control (TPM2_C3SC)	TPM2
4003_A028	Channel (n) Value (TPM2_C3V)	TPM2
4003_A02C	Channel (n) Status and Control (TPM2_C4SC)	TPM2
4003_A030	Channel (n) Value (TPM2_C4V)	TPM2
4003_A034	Channel (n) Status and Control (TPM2_C5SC)	TPM2
4003_A038	Channel (n) Value (TPM2_C5V)	TPM2

## 29.1.1 TPM instantiation information

This device contains three low power TPM modules (TPM). All TPM modules in the device are configured only as basic TPM function, do not support quadrature decoder function, and all can be functional in Stop/VLPS mode. The clock source is either external or internal in Stop/VLPS mode.

The following table shows how these modules are configured.

**Table 29-2. TPM configuration**

TPM instance	Number of channels	Features/usage
TPM0	6	Basic TPM, functional in Stop/VLPS mode
TPM1	2	Basic TPM, functional in Stop/VLPS mode
TPM2	2	Basic TPM, functional in Stop/VLPS mode

There are several connections to and from the TPMs in order to facilitate customer use cases. For complete details on the TPM module interconnects please refer to the [Module-to-Module section](#).

## 29.1.2 Clock options

The TPM blocks are clocked from a single TPM clock that can be selected from OSCERCLK, MCGIRCLK, or MCGPCLK. The selected source is controlled by SIM\_SOPT2[TPMSRC] .

Each TPM also supports an external clock mode (TPM\_SC[CMOD]=1x) in which the counter increments after a synchronized (to the selected TPM clock source) rising edge detect of an external clock input. The available external clock (either TPM\_CLKIN0 or TPM\_CLKIN1) is selected by SIM\_SOPT4[TPMxCLKSEL] control register. To guarantee valid operation the selected external clock must be less than half the frequency of the selected TPM clock source.

### 29.1.3 Trigger options

Each TPM has a selectable external trigger input source controlled by `TPMx_CONF[TRGSEL]` to use for starting the counter and/or reloading the counter. The options available are shown in the following table.

**Table 29-3. TPM external trigger options**

TPMx_CONF[TRGSEL]	Selected source
0000	External trigger pin input (EXTRG_IN)
0001	CMP0 output
0010	Reserved
0011	Reserved
0100	PIT trigger 0
0101	PIT trigger 1
0110	Reserved
0111	Reserved
1000	TPM0 overflow
1001	TPM1 overflow
1010	TPM2 overflow
1011	Reserved
1100	RTC alarm
1101	RTC seconds
1110	LPTMR trigger
1111	Reserved

### 29.1.4 Global timebase

Each TPM has a global timebase feature controlled by `TPMx_CONF[GTBEEN]`. TPM1 is configured as the global time when this option is enabled.

### 29.1.5 TPM interrupts

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

## 29.2 Introduction

The TPM (Timer/PWM Module) is a 2- to 8-channel timer which supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications.

The counter, compare and capture registers are clocked by an asynchronous clock that can remain enabled in low power modes. An example of using the TPM with the asynchronous DMA is described in [AN4631:Using the Asynchronous DMA features of the Kinetis L Series](#) .

### 29.2.1 TPM Philosophy

The TPM is built upon a very simple timer (HCS08 Timer PWM Module – TPM) used for many years on Freescale's 8-bit microcontrollers. The TPM extends the functionality to support operation in low power modes by clocking the counter, compare and capture registers from an asynchronous clock that can remain functional in low power modes.

### 29.2.2 Features

The TPM features include:

- TPM clock mode is selectable
  - Can increment on every edge of the asynchronous counter clock
  - Can increment on rising edge of an external clock input synchronized to the asynchronous counter clock
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128
- TPM includes a 16-bit counter
  - It can be a free-running counter or modulo counter
  - The counting can be up or up-down
- Includes 6 channels that can be configured for input capture, output compare, edge-aligned PWM mode, or center-aligned PWM mode
  - In input capture mode the capture can occur on rising edges, falling edges or both edges



- In output compare mode the output signal can be set, cleared, pulsed, or toggled on match
- All channels can be configured for edge-aligned PWM mode or center-aligned PWM mode
- Support the generation of an interrupt and/or DMA request per channel
- Support the generation of an interrupt and/or DMA request when the counter overflows
- Support selectable trigger input to optionally reset or cause the counter to start incrementing.
  - The counter can also optionally stop incrementing on counter overflow
- Support the generation of hardware triggers when the counter overflows and per channel

### 29.2.3 Modes of operation

During debug mode, the TPM can be configured to temporarily pause all counting until the core returns to normal user operating mode or to operate normally. When the counter is paused, trigger inputs and input capture events are ignored.

During doze mode, the TPM can be configured to operate normally or to pause all counting for the duration of doze mode. When the counter is paused, trigger inputs and input capture events are ignored.

During stop mode, the TPM counter clock can remain functional and the TPM can generate an asynchronous interrupt to exit the MCU from stop mode.

### 29.2.4 Block diagram

The TPM uses one input/output (I/O) pin per channel, CH<sub>n</sub> (TPM channel (n)) where n is the channel number.

The following figure shows the TPM structure. The central component of the TPM is the 16-bit counter with programmable final value and its counting can be up or up-down.

## TPM Signal Descriptions

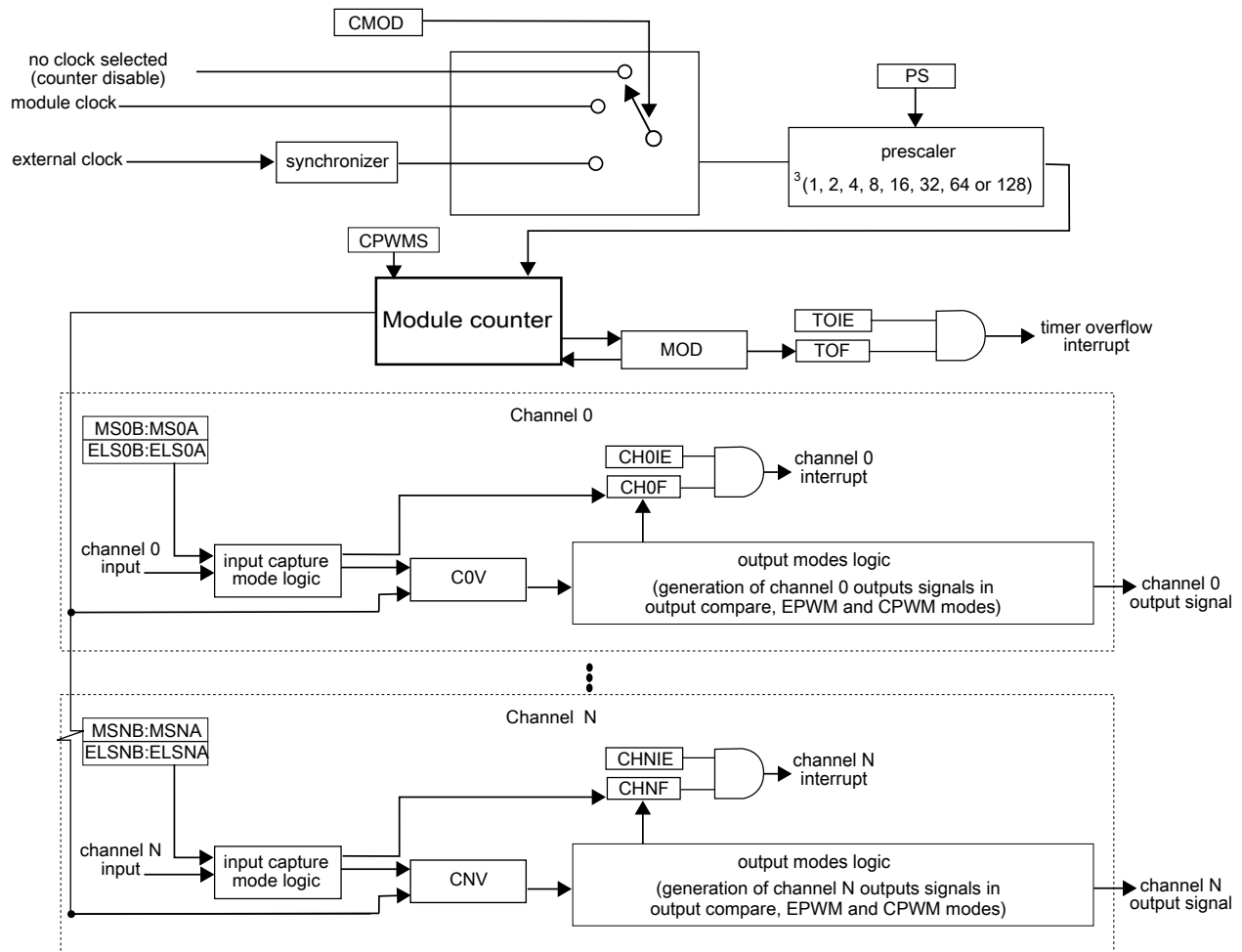


Figure 29-1. TPM block diagram

## 29.3 TPM Signal Descriptions

Table 29-4. TPM signal descriptions

Signal	Description	I/O
TPM_EXTCLK	External clock. TPM external clock can be selected to increment the TPM counter on every rising edge synchronized to the counter clock.	I
TPM_CHn	TPM channel (n = 5 to 0). A TPM channel pin is configured as output when configured in an output compare or PWM mode and the TPM counter is enabled, otherwise the TPM channel pin is an input.	I/O

### 29.3.1 TPM\_EXTCLK — TPM External Clock

The rising edge of the external input signal is used to increment the TPM counter if selected by CMOD[1:0] bits in the SC register. This input signal must be less than half of the TPM counter clock frequency. The TPM counter prescaler selection and settings are also used when an external input is selected.

### 29.3.2 TPM\_CHn — TPM Channel (n) I/O Pin

Each TPM 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.

## 29.4 Memory Map and Register Definition

This section provides a detailed description of all TPM registers.

Attempting to access a reserved register location in the TPM memory map will generate a bus error.

TPM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_8000	Status and Control (TPM0_SC)	32	R/W	0000_0000h	<a href="#">29.4.1/453</a>
4003_8004	Counter (TPM0_CNT)	32	R/W	0000_0000h	<a href="#">29.4.2/454</a>
4003_8008	Modulo (TPM0_MOD)	32	R/W	0000_FFFFh	<a href="#">29.4.3/455</a>
4003_800C	Channel (n) Status and Control (TPM0_C0SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_8010	Channel (n) Value (TPM0_C0V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_8014	Channel (n) Status and Control (TPM0_C1SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_8018	Channel (n) Value (TPM0_C1V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_801C	Channel (n) Status and Control (TPM0_C2SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_8020	Channel (n) Value (TPM0_C2V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_8024	Channel (n) Status and Control (TPM0_C3SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_8028	Channel (n) Value (TPM0_C3V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_802C	Channel (n) Status and Control (TPM0_C4SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_8030	Channel (n) Value (TPM0_C4V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_8034	Channel (n) Status and Control (TPM0_C5SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_8038	Channel (n) Value (TPM0_C5V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_8050	Capture and Compare Status (TPM0_STATUS)	32	R/W	0000_0000h	<a href="#">29.4.6/458</a>

Table continues on the next page...

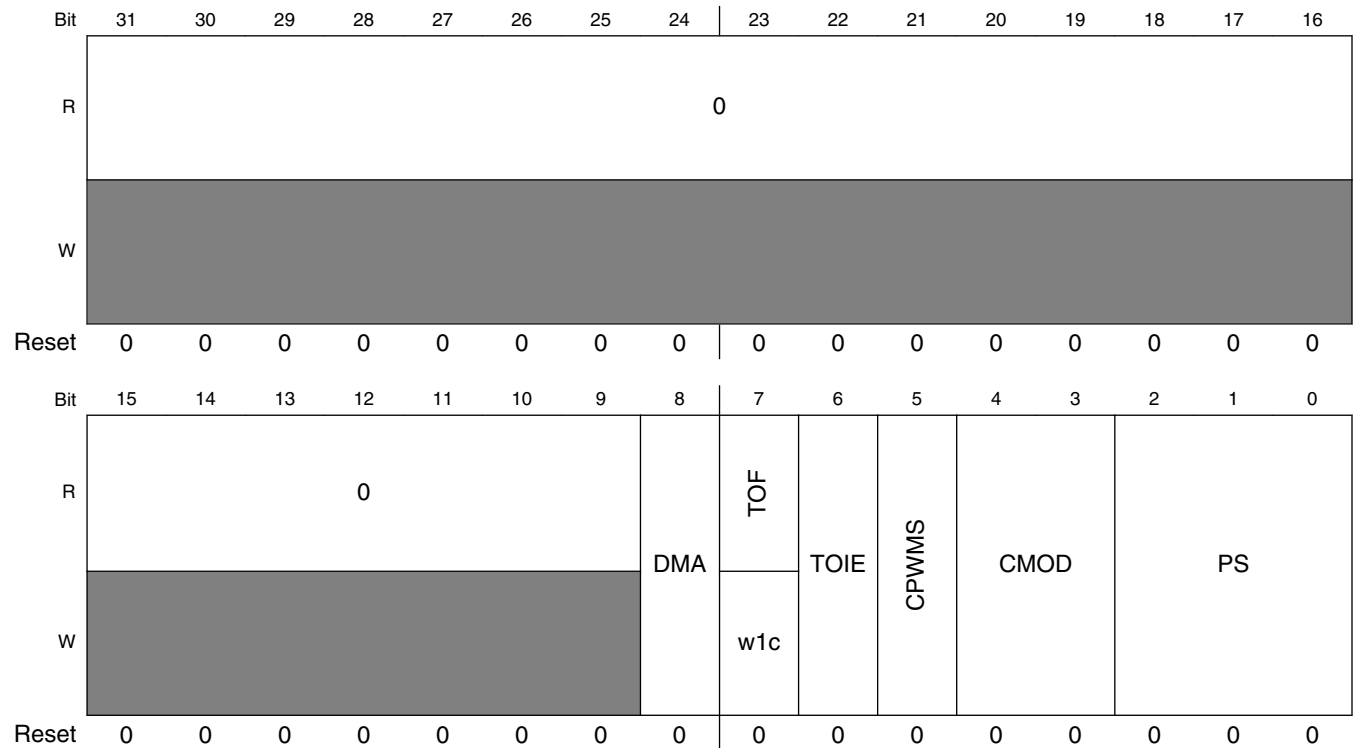
## TPM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_8070	Channel Polarity (TPM0_POL)	32	R/W	0000_0000h	<a href="#">29.4.7/460</a>
4003_8084	Configuration (TPM0_CONF)	32	R/W	0000_0000h	<a href="#">29.4.8/461</a>
4003_9000	Status and Control (TPM1_SC)	32	R/W	0000_0000h	<a href="#">29.4.1/453</a>
4003_9004	Counter (TPM1_CNT)	32	R/W	0000_0000h	<a href="#">29.4.2/454</a>
4003_9008	Modulo (TPM1_MOD)	32	R/W	0000_FFFFh	<a href="#">29.4.3/455</a>
4003_900C	Channel (n) Status and Control (TPM1_C0SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_9010	Channel (n) Value (TPM1_C0V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_9014	Channel (n) Status and Control (TPM1_C1SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_9018	Channel (n) Value (TPM1_C1V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_901C	Channel (n) Status and Control (TPM1_C2SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_9020	Channel (n) Value (TPM1_C2V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_9024	Channel (n) Status and Control (TPM1_C3SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_9028	Channel (n) Value (TPM1_C3V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_902C	Channel (n) Status and Control (TPM1_C4SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_9030	Channel (n) Value (TPM1_C4V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_9034	Channel (n) Status and Control (TPM1_C5SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_9038	Channel (n) Value (TPM1_C5V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_9050	Capture and Compare Status (TPM1_STATUS)	32	R/W	0000_0000h	<a href="#">29.4.6/458</a>
4003_9070	Channel Polarity (TPM1_POL)	32	R/W	0000_0000h	<a href="#">29.4.7/460</a>
4003_9084	Configuration (TPM1_CONF)	32	R/W	0000_0000h	<a href="#">29.4.8/461</a>
4003_A000	Status and Control (TPM2_SC)	32	R/W	0000_0000h	<a href="#">29.4.1/453</a>
4003_A004	Counter (TPM2_CNT)	32	R/W	0000_0000h	<a href="#">29.4.2/454</a>
4003_A008	Modulo (TPM2_MOD)	32	R/W	0000_FFFFh	<a href="#">29.4.3/455</a>
4003_A00C	Channel (n) Status and Control (TPM2_C0SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_A010	Channel (n) Value (TPM2_C0V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_A014	Channel (n) Status and Control (TPM2_C1SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_A018	Channel (n) Value (TPM2_C1V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_A01C	Channel (n) Status and Control (TPM2_C2SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_A020	Channel (n) Value (TPM2_C2V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_A024	Channel (n) Status and Control (TPM2_C3SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_A028	Channel (n) Value (TPM2_C3V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_A02C	Channel (n) Status and Control (TPM2_C4SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_A030	Channel (n) Value (TPM2_C4V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_A034	Channel (n) Status and Control (TPM2_C5SC)	32	R/W	0000_0000h	<a href="#">29.4.4/456</a>
4003_A038	Channel (n) Value (TPM2_C5V)	32	R/W	0000_0000h	<a href="#">29.4.5/458</a>
4003_A050	Capture and Compare Status (TPM2_STATUS)	32	R/W	0000_0000h	<a href="#">29.4.6/458</a>
4003_A070	Channel Polarity (TPM2_POL)	32	R/W	0000_0000h	<a href="#">29.4.7/460</a>
4003_A084	Configuration (TPM2_CONF)	32	R/W	0000_0000h	<a href="#">29.4.8/461</a>

### 29.4.1 Status and Control (TPMx\_SC)

SC contains the overflow status flag and control bits used to configure the interrupt enable, module configuration and prescaler factor. These controls relate to all channels within this module.

Address: Base address + 0h offset



**TPMx\_SC field descriptions**

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 DMA	DMA Enable Enables DMA transfers for the overflow flag. 0 Disables DMA transfers. 1 Enables DMA transfers.
7 TOF	Timer Overflow Flag Set by hardware when the TPM counter equals the value in the MOD register and increments. Writing a 1 to TOF clears it. Writing a 0 to TOF has no effect. If another TPM overflow occurs between the flag setting and the flag clearing, the write operation has no effect; therefore, TOF remains set indicating another overflow has occurred. In this case a TOF interrupt request is not lost due to a delay in clearing the previous TOF.

*Table continues on the next page...*

## TPMx\_SC field descriptions (continued)

Field	Description
	0 TPM counter has not overflowed. 1 TPM counter has overflowed.
6 TOIE	Timer Overflow Interrupt Enable Enables TPM overflow interrupts.  0 Disable TOF interrupts. Use software polling or DMA request. 1 Enable TOF interrupts. An interrupt is generated when TOF equals one.
5 CPWMS	Center-Aligned PWM Select Selects CPWM mode. This mode configures the TPM to operate in up-down counting mode. This field is write protected. It can be written only when the counter is disabled.  0 TPM counter operates in up counting mode. 1 TPM counter operates in up-down counting mode.
4–3 CMOD	Clock Mode Selection Selects the TPM counter clock modes. When disabling the counter, this field remain set until acknowledged in the TPM clock domain.  00 TPM counter is disabled 01 TPM counter increments on every TPM counter clock 10 TPM counter increments on rising edge of TPM_EXTCLK synchronized to the TPM counter clock 11 Reserved.
PS	Prescale Factor Selection Selects one of 8 division factors for the clock mode selected by CMOD. This field is write protected. It can be written only when the counter is disabled.  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

## 29.4.2 Counter (TPMx\_CNT)

The CNT register contains the TPM counter value.

Reset clears the CNT register. Writing any value to COUNT also clears the counter.

When debug is active, the TPM counter does not increment unless configured otherwise.

Reading the CNT register adds two wait states to the register access due to synchronization delays.

Address: Base address + 4h offset

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

**TPMx\_CNT field descriptions**

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Counter value

**29.4.3 Modulo (TPMx\_MOD)**

The Modulo register contains the modulo value for the TPM counter. When the TPM counter reaches the modulo value and increments, the overflow flag (TOF) is set and the next value of TPM 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 [MOD Register Update](#) . Additional writes to the MOD write buffer are ignored until the register has been updated.

It is recommended to initialize the TPM counter (write to CNT) before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

Address: Base address + 8h offset

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

**TPMx\_MOD field descriptions**

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MOD	Modulo value  This field must be written with single 16-bit or 32-bit access.

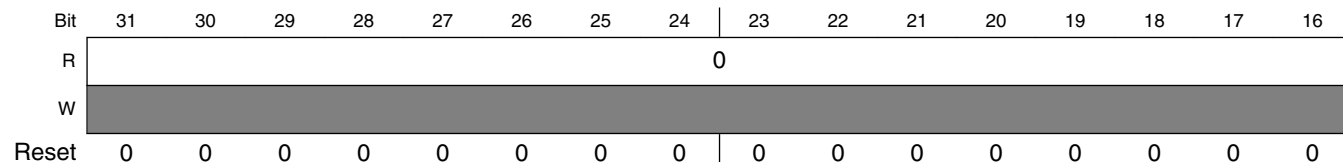
### 29.4.4 Channel (n) Status and Control (TPMx\_CnSC)

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function. When switching from one channel mode to a different channel mode, the channel must first be disabled and this must be acknowledged in the TPM counter clock domain.

**Table 29-5. Mode, Edge, and Level Selection**

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	00	00	None	Channel disabled
X	01	00	Software compare	Pin not used for TPM
0	00	01	Input capture	Capture on Rising Edge Only
		10		Capture on Falling Edge Only
		11		Capture on Rising or Falling Edge
	01	01	Output compare	Toggle Output on match
		10		Clear Output on match
		11		Set Output on match
	10	10	Edge-aligned PWM	High-true pulses (clear Output on match, set Output on reload)
				Low-true pulses (set Output on match, clear Output on reload)
	11	10	Output compare	Pulse Output low on match
				01
1	10	10	Center-aligned PWM	High-true pulses (clear Output on match-up, set Output on match-down)
				01

Address: Base address + Ch offset + (8d × i), where i=0d to 5d





Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								CHF	CHIE	MSB	MSA	ELSB	ELSA	0	DMA
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### TPMx\_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 writing a 1 to the CHF bit. Writing a 0 to CHF has no effect.  If another event occurs between the CHF sets and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the delay in clearing the 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. 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. When a channel is disabled, this field will not change state until acknowledged in the TPM counter clock domain.
4 MSA	Channel Mode Select  Used for further selections in the channel logic. Its functionality is dependent on the channel mode. When a channel is disabled, this field will not change state until acknowledged in the TPM counter clock domain.
3 ELSB	Edge or Level Select  The functionality of ELSB and ELSA depends on the channel mode. When a channel is disabled, this field will not change state until acknowledged in the TPM counter clock domain.
2 ELSA	Edge or Level Select  The functionality of ELSB and ELSA depends on the channel mode. When a channel is disabled, this field will not change state until acknowledged in the TPM counter clock domain.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 DMA	DMA Enable  Enables DMA transfers for the channel.  0 Disable DMA transfers. 1 Enable DMA transfers.

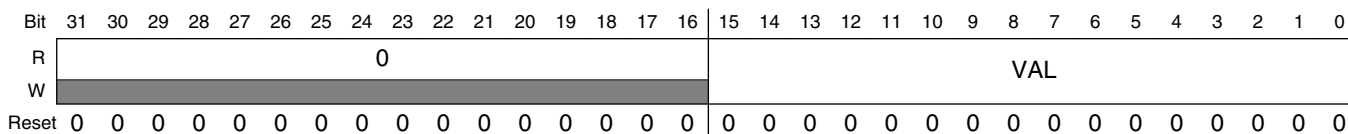
### 29.4.5 Channel (n) Value (TPMx\_CnV)

These registers contain the captured TPM counter value for the input modes or the match value for the output modes.

In input capture mode, any write to a CnV register is ignored.

In compare 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 [CnV Register Update](#) . Additional writes to the CnV write buffer are ignored until the register has been updated.

Address: Base address + 10h offset + (8d × i), where i=0d to 5d



**TPMx\_CnV field descriptions**

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
VAL	Channel Value  Captured TPM counter value of the input modes or the match value for the output modes. This field must be written with single 16-bit or 32-bit access.

### 29.4.6 Capture and Compare Status (TPMx\_STATUS)

The STATUS register contains a copy of the status flag, CnSC[CHnF] for each TPM channel, as well as SC[TOF], 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 writing all ones to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. Writing a 1 to CHF clears it. Writing a 0 to CHF has no effect.

If another event occurs between the flag setting and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF.

Address: Base address + 50h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	0																
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	0								TOF	0		CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W									w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**TPMx\_STATUS field descriptions**

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 TOF	Timer Overflow Flag See register description 0 TPM counter has not overflowed. 1 TPM counter has overflowed.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
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.

Table continues on the next page...

**TPMx\_STATUS field descriptions (continued)**

Field	Description
2 CH2F	Channel 2 Flag  See the register 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.

**29.4.7 Channel Polarity (TPMx\_POL)**

This register defines the input and output polarity of each of the channels.

Address: Base address + 70h 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								POL5	POL4	POL3	POL2	POL1	POL0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**TPMx\_POL field descriptions**

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 POL5	Channel 5 Polarity  0 The channel polarity is active high. 1 The channel polarity is active low.
4 POL4	Channel 4 Polarity  0 The channel polarity is active high 1 The channel polarity is active low.

*Table continues on the next page...*

### TPMx\_POL field descriptions (continued)

Field	Description
3 POL3	Channel 3 Polarity 0 The channel polarity is active high. 1 The channel polarity is active low.
2 POL2	Channel 2 Polarity 0 The channel polarity is active high. 1 The channel polarity is active low.
1 POL1	Channel 1 Polarity 0 The channel polarity is active high. 1 The channel polarity is active low.
0 POL0	Channel 0 Polarity 0 The channel polarity is active high. 1 The channel polarity is active low.

### 29.4.8 Configuration (TPMx\_CONF)

This register selects the behavior in debug and wait modes and the use of an external global time base.

Address: Base address + 84h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				TRGSEL				TRGSRC	TRGPOL	0		CPOT	CROT	CSOO	CSOT
W	[Shaded]				[Shaded]				[Shaded]	[Shaded]	[Shaded]		[Shaded]	[Shaded]	[Shaded]	[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				GTBEEN	GTBSYNC	DBGMODE		DOZEEN	0						
W	[Shaded]				[Shaded]	[Shaded]	[Shaded]		[Shaded]	[Shaded]						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### TPMx\_CONF field descriptions

Field	Description
31–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

## TPMx\_CONF field descriptions (continued)

Field	Description
27–24 TRGSEL	<p>Trigger Select</p> <p>Selects the input trigger to use for starting, reloading and/or pausing the counter. The source of the trigger (external or internal to the TPM) is configured by the TRGSRC field. This field should only be changed when the TPM counter is disabled.</p> <p>Refer to the chip configuration section for available external trigger options.</p> <p>The available internal trigger sources are listed below.</p> <p>0001 Channel 0 pin input capture  0010 Channel 1 pin input capture  0011 Channel 0 or Channel 1 pin input capture  0100 Channel 2 pin input capture  0101 Channel 0 or Channel 2 pin input capture  0110 Channel 1 or Channel 2 pin input capture  0111 Channel 0 or Channel 1 or Channel 2 pin input capture  1000 Channel 3 pin input capture  1001 Channel 0 or Channel 3 pin input capture  1010 Channel 1 or Channel 3 pin input capture  1011 Channel 0 or Channel 1 or Channel 3 pin input capture  1100 Channel 2 or Channel 3 pin input capture  1101 Channel 0 or Channel 2 or Channel 3 pin input capture  1110 Channel 1 or Channel 2 or Channel 3 pin input capture  1111 Channel 0 or Channel 1 or Channel 2 or Channel 3 pin input capture</p>
23 TRGSRC	<p>Trigger Source</p> <p>Selects between internal (channel pin input capture) or external trigger sources.</p> <p>When selecting an internal trigger, the channel selected should be configured for input capture. Only a rising edge input capture can be used to initially start the counter using the CSOT configuration; either rising edge or falling edge input capture can be used to reload the counter using the CROT configuration; and the state of the channel input pin is used to pause the counter using the CPOT configuration. The channel polarity register can be used to invert the polarity of the channel input pins.</p> <p>This field should only be changed when the TPM counter is disabled.</p> <p>0 Trigger source selected by TRGSEL is external.  1 Trigger source selected by TRGSEL is internal (channel pin input capture).</p>
22 TRGPOL	<p>Trigger Polarity</p> <p>Selects the polarity of the external trigger source. This field should only be changed when the TPM counter is disabled.</p> <p>0 Trigger is active high.  1 Trigger is active low.</p>
21–20 Reserved	<p>This field is reserved.  This read-only field is reserved and always has the value 0.</p>
19 CPOT	<p>Counter Pause On Trigger</p> <p>When enabled, the counter will pause incrementing while the trigger remains asserted (level sensitive). This field should only be changed when the TPM counter is disabled.</p>
18 CROT	<p>Counter Reload On Trigger</p>

Table continues on the next page...

## TPMx\_CONF field descriptions (continued)

Field	Description
	<p>When set, the TPM counter will reload with 0 (and initialize PWM outputs to their default value) when a rising edge is detected on the selected trigger input.</p> <p>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.</p> <p>0 Counter is not reloaded due to a rising edge on the selected input trigger 1 Counter is reloaded when a rising edge is detected on the selected input trigger</p>
17 CSOO	<p>Counter Stop On Overflow</p> <p>When set, the TPM counter will stop incrementing once the counter equals the MOD value and incremented (this also sets the TOF). Reloading the counter with 0 due to writing to the counter register or due to a trigger input does not cause the counter to stop incrementing. Once the counter has stopped incrementing, the counter will not start incrementing unless it is disabled and then enabled again, or a rising edge on the selected trigger input is detected when CSOT set.</p> <p>This field should only be changed when the TPM counter is disabled.</p> <p>0 TPM counter continues incrementing or decrementing after overflow 1 TPM counter stops incrementing or decrementing after overflow.</p>
16 CSOT	<p>Counter Start on Trigger</p> <p>When set, the TPM counter will not start incrementing after it is enabled until a rising edge on the selected trigger input is detected. If the TPM counter is stopped due to an overflow, a rising edge on the selected trigger input will also cause the TPM counter to start incrementing again.</p> <p>The trigger input is ignored if the TPM counter is paused during debug mode or doze mode. This field should only be changed when the TPM counter is disabled.</p> <p>0 TPM counter starts to increment immediately, once it is enabled. 1 TPM counter only starts to increment when it a rising edge on the selected input trigger is detected, after it has been enabled or after it has stopped due to overflow.</p>
15–10 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
9 GTBEEN	<p>Global time base enable</p> <p>Configures the TPM to use an externally generated global time base counter. When an externally generated timebase is used, the internal TPM counter is not used by the channels but can be used to generate a periodic interruptor DMA request using the Modulo register and timer overflow flag.</p> <p>0 All channels use the internally generated TPM counter as their timebase 1 All channels use an externally generated global timebase as their timebase</p>
8 GTBSYNC	<p>Global Time Base Synchronization</p> <p>When enabled, the TPM counter is synchronized to the global time base. It uses the global timebase enable, trigger and overflow to ensure the TPM counter starts incrementing at the same time as the global timebase, stops incrementing at the same time as the global timebase and is reset at the same time as the global timebase. This field should only be changed when the TPM counter is disabled.</p> <p>0 Global timebase synchronization disabled. 1 Global timebase synchronization enabled.</p>
7–6 DBGMODE	<p>Debug Mode</p> <p>Configures the TPM behavior in debug mode. All other configurations are reserved.</p>

*Table continues on the next page...*

### TPM<sub>x</sub>\_CONF field descriptions (continued)

Field	Description
	00 TPM counter is paused and does not increment during debug mode. Trigger inputs and input capture events are also ignored. 11 TPM counter continues in debug mode.
5 DOZEEN	Doze Enable  Configures the TPM behavior in wait mode.  0 Internal TPM counter continues in Doze mode. 1 Internal TPM counter is paused and does not increment during Doze mode. Trigger inputs and input capture events are also ignored.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

## 29.5 Functional description

The following sections describe the TPM features.

### 29.5.1 Clock domains

The TPM module supports two clock domains.

The bus clock domain is used by the register interface and for synchronizing interrupts and DMA requests.

The TPM counter clock domain is used to clock the counter and prescaler along with the output compare and input capture logic. The TPM counter clock is considered asynchronous to the bus clock, can be a higher or lower frequency than the bus clock and can remain operational in Stop mode. Multiple TPM instances are all clocked by the same TPM counter clock in support of the external timebase feature.

#### 29.5.1.1 Counter Clock Mode

The CMOD[1:0] bits in the SC register either disable the TPM counter or select one of two possible clock modes for the TPM counter. After any reset, CMOD[1:0] = 0:0 so the TPM counter is disabled.

The CMOD[1:0] bits may be read or written at any time. Disabling the TPM counter by writing zero to the CMOD[1:0] bits does not affect the TPM counter value or other registers, but must be acknowledged by the TPM counter clock domain before they read as zero.

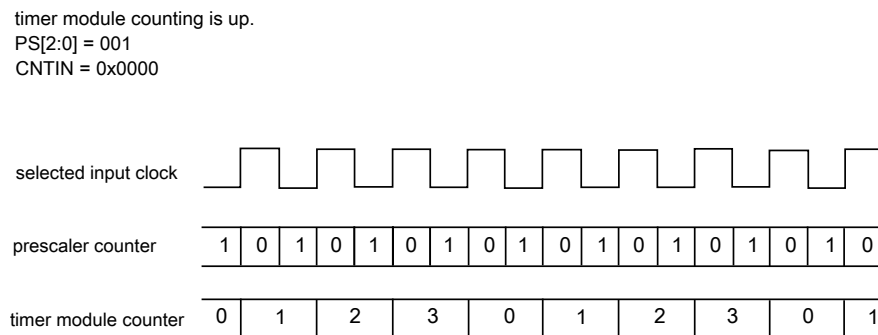


The external clock input passes through a synchronizer clocked by the TPM counter clock to assure that counter transitions are properly aligned to counter clock transitions. Therefore, to meet Nyquist criteria considering also jitter, the frequency of the external clock source must be less than half of the counter clock frequency.

## 29.5.2 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 TPM counter.



**Figure 29-2. Example of the Prescaler Counter**

## 29.5.3 Counter

The TPM has a 16-bit counter that is used by the channels either for input or output modes.

The counter updates from the selected clock divided by the prescaler.

The TPM counter has these modes of operation:

- up counting (see [Up counting](#))
- up-down counting (see [Up-down counting](#))

### 29.5.3.1 Up counting

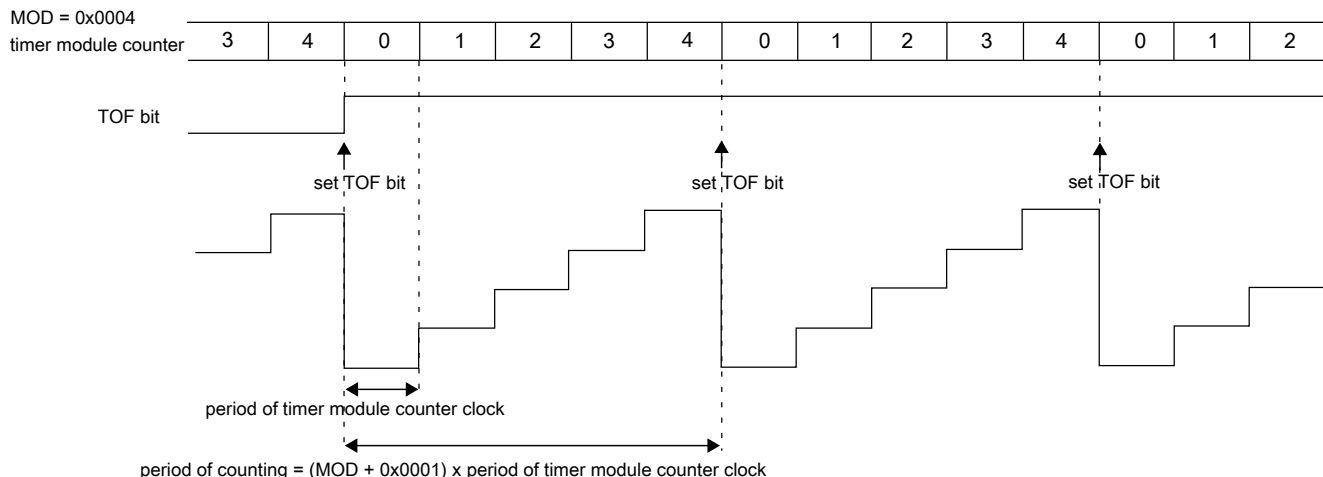
Up counting is selected when SC[CPWMS] = 0.

The value of zero is loaded into the TPM counter, and the counter increments until the value of MOD is reached, at which point the counter is reloaded with zero.

## Functional description

The TPM period when using up counting is  $(MOD + 0x0001) \times$  period of the TPM counter clock.

The TOF bit is set when the TPM counter changes from MOD to zero.



**Figure 29-3. Example of TPM Up Counting**

### Note

- MOD = 0000 is a redundant condition. In this case, the TPM counter is always equal to MOD and the TOF bit is set in each rising edge of the TPM counter clock.

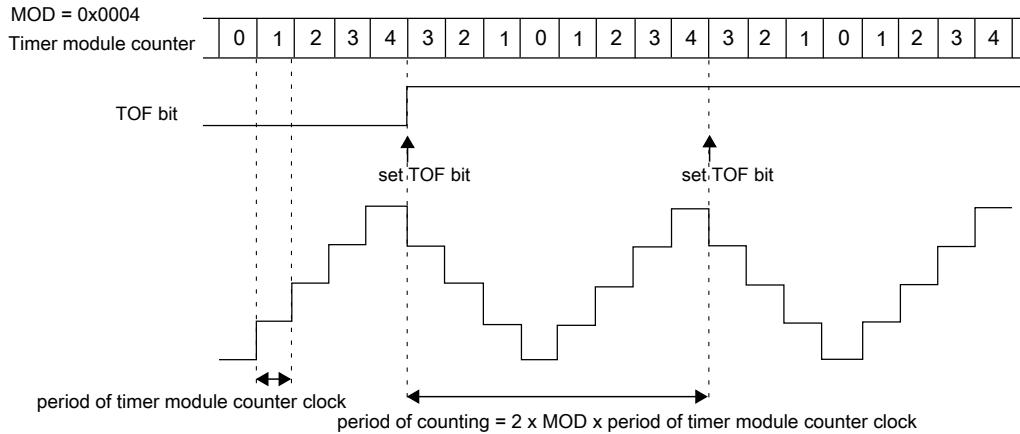
### 29.5.3.2 Up-down counting

Up-down counting is selected when  $SC[CPWMS] = 1$ . When configured for up-down counting, configuring  $CONF[MOD]$  to less than 2 is not supported.

The value of 0 is loaded into the TPM counter, and the counter increments until the value of MOD is reached, at which point the counter is decremented until it returns to zero and the up-down counting restarts.

The TPM period when using up-down counting is  $2 \times MOD \times$  period of the TPM counter clock.

The TOF bit is set when the TPM counter changes from MOD to  $(MOD - 1)$ .



**Figure 29-4. Example of up-down counting**

### 29.5.3.3 Counter Reset

Any write to CNT resets the TPM counter and the channel outputs to their initial values (except for channels in output compare mode).

### 29.5.3.4 Global time base (GTB)

The global time base (GTB) is a TPM function that allows multiple TPM modules to share the same timebase. When the global time base is enabled ( $\text{CONF}[\text{GTBEEN}] = 1$ ), the local TPM channels use the counter value, counter enable and overflow indication from the TPM generating the global time base. If the local TPM counter is not generating the global time base, then it can be used as an independent counter or pulse accumulator.

The local TPM counter can also be configured to synchronize to the global time base, by configuring ( $\text{GTBSYNC} = 1$ ). When synchronized to the global time base, the local counter will use the counter enable and counter overflow indication from the TPM generating the global time base. This enables multiple TPM to be configured with the same phase, but with different periods (although the global time base must be configured with the longest period).

### 29.5.3.5 Counter trigger

The TPM counter can be configured to start, stop or reset in response to a hardware trigger input. The trigger input is synchronized to the asynchronous counter clock, so there is a 3 counter clock delay between the trigger assertion and the counter responding.

## Functional description

- When (CSOT = 1), the counter will not start incrementing until a rising edge is detected on the trigger input.
- When (CSOO= 1), the counter will stop incrementing whenever the TOF flag is set. The counter does not increment again unless it is disabled, or if CSOT = 1 and a rising edge is detected on the trigger input.
- When (CROT= 1), the counter will reset to zero as if an overflow occurred whenever a rising edge is detected on the trigger input.
- When (CPOT = 1), the counter will pause incrementing whenever the trigger input is asserted. The counter will continue incrementing when the trigger input negates.

The polarity of the external input trigger can be configured by the TRGPOL register bit.

When an internal trigger source is selected, the trigger input is selected from one or more channel input capture events. The input capture filters are used with the internal trigger sources and the POLn bits can be used to invert the polarity of the input channels. Note that following restrictions apply with input capture channel sources.

- When (CSOT = 1), the counter will only start incrementing on a rising edge on the channel input, provided ELSnA = 1.
- When (CROT= 1), the counter will reset to zero on either edge of the channel input, as configured by ELSnB:ELSnA.
- When (CPOT = 1), the counter will pause incrementing whenever the channel input is asserted.

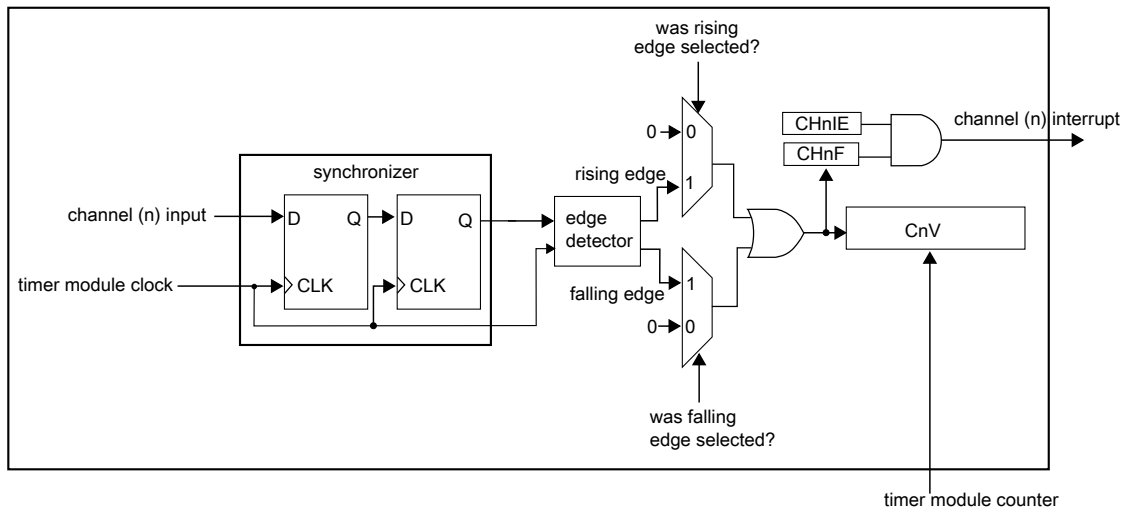
### 29.5.4 Input Capture Mode

The input capture mode is selected when (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 TPM 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 TPM\_CHn 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 counter clock divided by 4, which is required to meet Nyquist criteria for signal sampling.

Writes to the CnV register are ignored in input capture mode.



**Figure 29-5. Input capture mode**

The CHnF bit is set on the third rising edge of the counter clock after a valid edge occurs on the channel input.

### 29.5.5 Output Compare Mode

The output compare mode is selected when (CPWMS = 0), and (MSnB:MSnA = X:1).

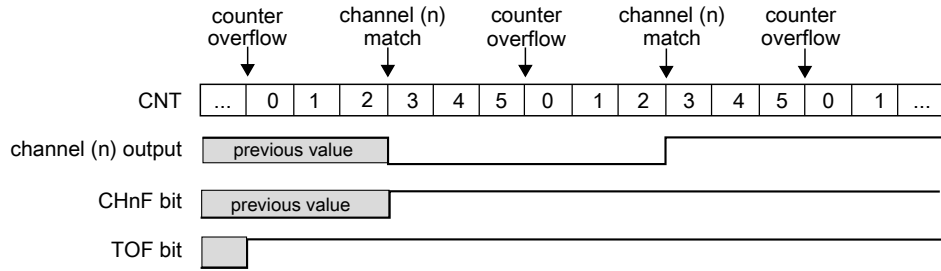
In output compare mode, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CnV register of an output compare channel, the channel (n) output can be set, cleared or toggled if MSnB is clear. If MSnB is set then the channel (n) output is pulsed high or low for as long as the counter matches the value in the CnV register.

When a channel is initially configured to output compare mode, the channel output updates with its negated value (logic 0 for set/toggle/pulse high and logic one for clear/pulse low).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (TPM counter = CnV).

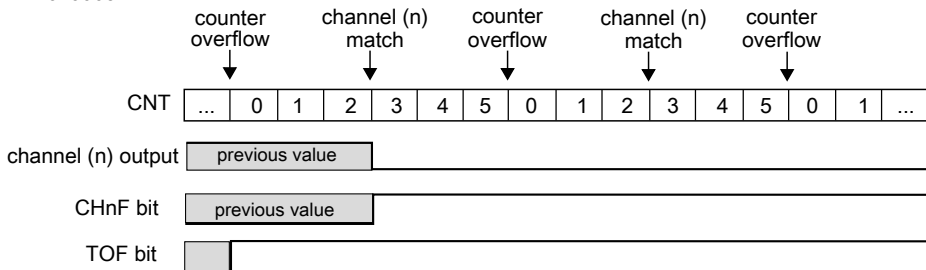
## Functional description

MOD = 0x0005  
CnV = 0x0003



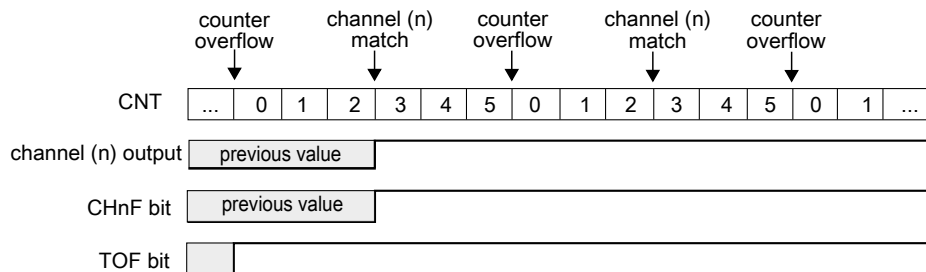
**Figure 29-6. Example of the output compare mode when the match toggles the channel output**

MOD = 0x0005  
CnV = 0x0003



**Figure 29-7. Example of the output compare mode when the match clears the channel output**

MOD = 0x0005  
CnV = 0x0003



**Figure 29-8. Example of the output compare mode when the match sets the channel output**

It is possible to use the output compare mode with (ELSnB:ELSnA = 0:0). In this case, 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 TPM.

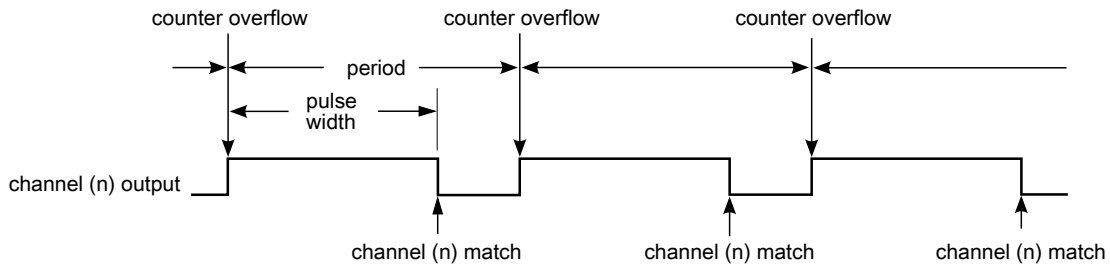
## 29.5.6 Edge-Aligned PWM (EPWM) Mode

The edge-aligned mode is selected when (CPWMS = 0), and (MSnB:MSnA = 1:0).

The EPWM period is determined by  $(MOD + 0x0001)$  and the pulse width (duty cycle) is determined by  $CnV$ .

The  $CHnF$  bit is set and the channel (n) interrupt is generated (if  $CHnIE = 1$ ) at the channel (n) match (TPM counter =  $CnV$ ), that is, at the end of the pulse width.

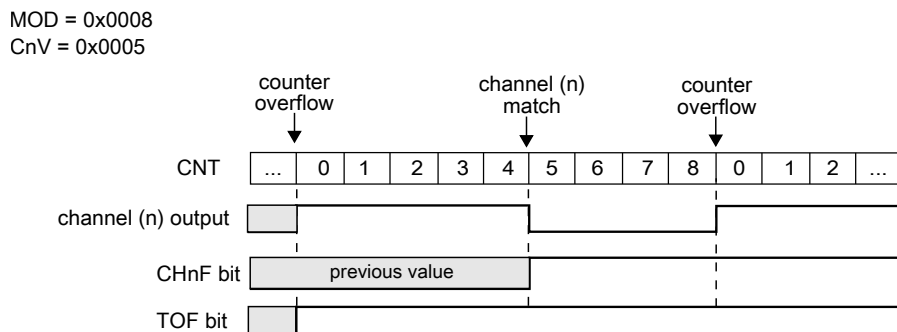
This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an TPM.



**Figure 29-9. 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 TPM.

If ( $ELSnB:ELSnA = 1:0$ ), then the channel (n) output is forced high at the counter overflow (when the zero is loaded into the TPM counter), and it is forced low at the channel (n) match (TPM counter =  $CnV$ ) (see the following figure).

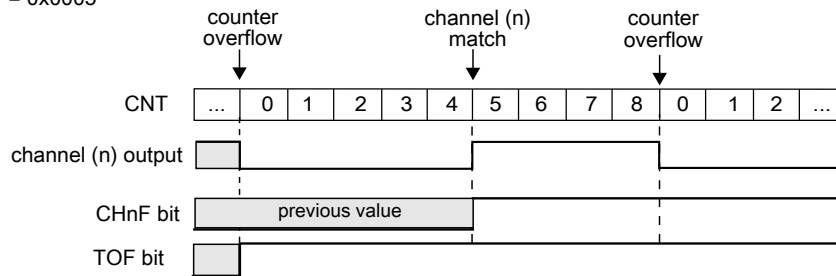


**Figure 29-10. EPWM signal with  $ELSnB:ELSnA = 1:0$**

If ( $ELSnB:ELSnA = X:1$ ), then the channel (n) output is forced low at the counter overflow (when zero is loaded into the TPM counter), and it is forced high at the channel (n) match (TPM counter =  $CnV$ ) (see the following figure).

## Functional description

MOD = 0x0008  
CnV = 0x0005



**Figure 29-11. EPWM signal with ELSnB:ELSnA = X:1**

If ( $CnV = 0x0000$ ), then the channel (n) output is a 0% duty cycle EPWM signal. If ( $CnV > MOD$ ), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set since there is never a channel (n) match. Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

### 29.5.7 Center-Aligned PWM (CPWM) Mode

The center-aligned mode is selected when ( $CPWMS = 1$ ) and ( $MSnB:MSnA = 1:0$ ).

The CPWM pulse width (duty cycle) is determined by  $2 \times CnV$  and the period is determined by  $2 \times MOD$  (see the following figure). MOD must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

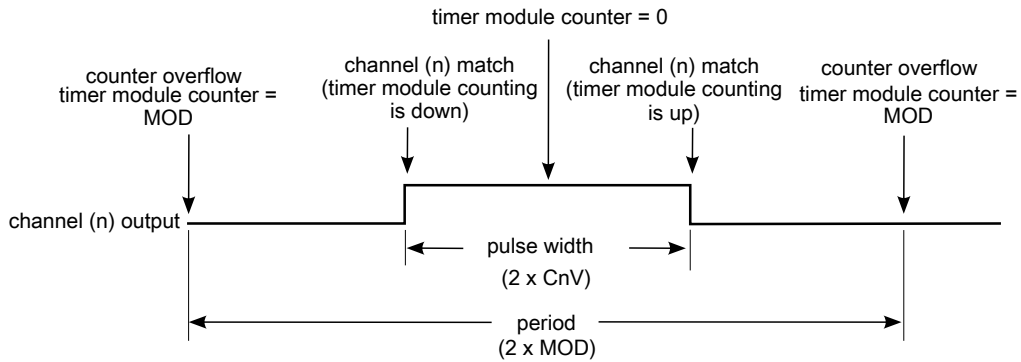
In the CPWM mode, the TPM counter counts up until it reaches MOD and then counts down until it reaches zero.

The CHnF bit is set and channel (n) interrupt is generated (if  $CHnIE = 1$ ) at the channel (n) match (TPM counter = CnV) when the TPM counting is down (at the begin of the pulse width) and when the TPM 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 when the TPM counter is zero.

The other channel modes are not designed to be used with the up-down counter ( $CPWMS = 1$ ). Therefore, all TPM channels should be used in CPWM mode when ( $CPWMS = 1$ ).

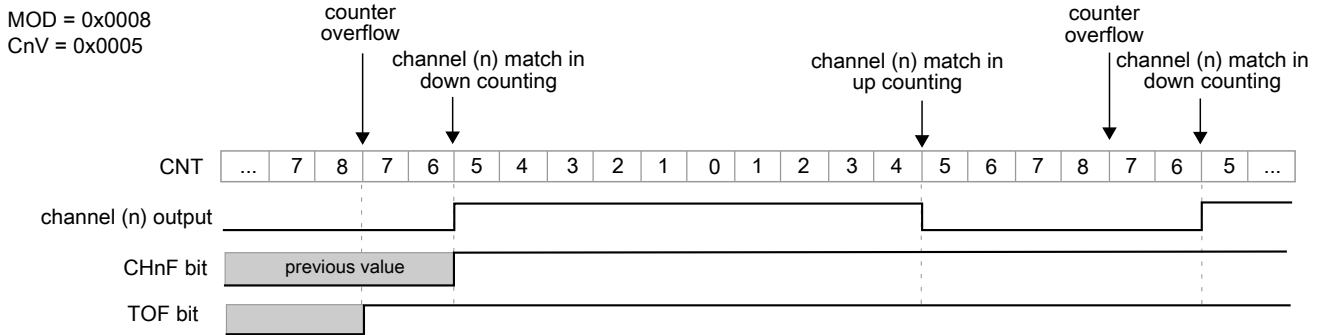




**Figure 29-12. CPWM period and pulse width with ELSnB:ELSnA = 1:0**

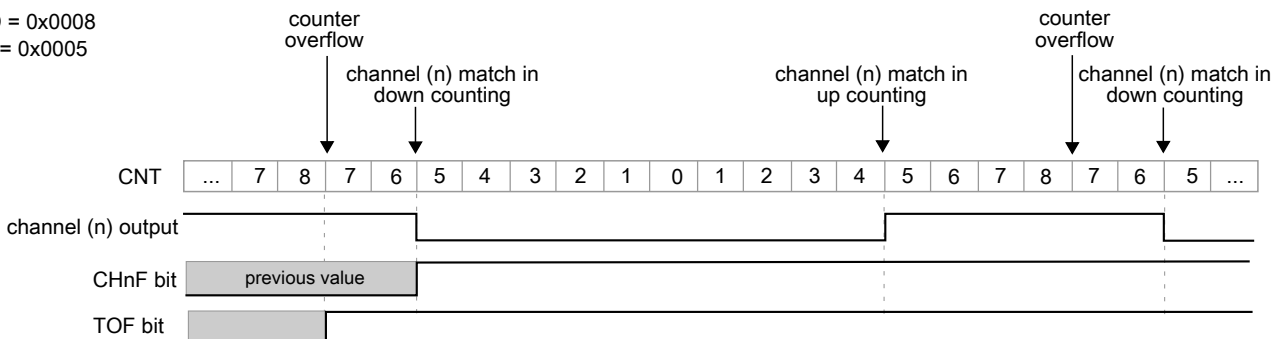
If (ELSnB:ELSnA = 0:0) when the TPM counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by TPM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the channel (n) match (TPM counter = CnV) when counting down, and it is forced low at the channel (n) match when counting up (see the following figure).



**Figure 29-13. 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 (TPM counter = CnV) when counting down, and it is forced high at the channel (n) match when counting up (see the following figure).



**Figure 29-14. CPWM signal with ELSnB:ELSnA = X:1**

If ( $CnV = 0x0000$ ) then the channel (n) output is a 0% duty cycle CPWM signal.

If ( $CnV > MOD$ ), then the channel (n) output is a 100% duty cycle CPWM signal, although the  $CHnF$  bit is set when the counter changes from incrementing to decrementing. Therefore,  $MOD$  must be less than  $0xFFFF$  in order to get a 100% duty cycle CPWM signal.

## 29.5.8 Registers Updated from Write Buffers

### 29.5.8.1 MOD Register Update

If ( $CMOD[1:0] = 0:0$ ) then  $MOD$  register is updated when  $MOD$  register is written.

If ( $CMOD[1:0] \neq 0:0$ ), then  $MOD$  register is updated according to the  $CPWMS$  bit, that is:

- If the selected mode is not CPWM then  $MOD$  register is updated after  $MOD$  register was written and the TPM counter changes from  $MOD$  to zero.
- If the selected mode is CPWM then  $MOD$  register is updated after  $MOD$  register was written and the TPM counter changes from  $MOD$  to  $(MOD - 1)$ .

### 29.5.8.2 CnV Register Update

If ( $CMOD[1:0] = 0:0$ ) then  $CnV$  register is updated when  $CnV$  register is written.

If ( $CMOD[1:0] \neq 0:0$ ), then  $CnV$  register is updated according to the selected mode, that is:

- If the selected mode is output compare then  $CnV$  register is updated on the next TPM counter increment (end of the prescaler counting) after  $CnV$  register was written.
- If the selected mode is EPWM then  $CnV$  register is updated after  $CnV$  register was written and the TPM counter changes from  $MOD$  to zero.
- If the selected mode is CPWM then  $CnV$  register is updated after  $CnV$  register was written and the TPM counter changes from  $MOD$  to  $(MOD - 1)$ .

## 29.5.9 DMA

The channel and overflow flags generate a DMA transfer request according to DMA and CHnIE/TOIE bits.

See the following table for more information.

**Table 29-6. DMA Transfer Request**

DMA	CHnIE/TOIE	Channel/Overflow DMA Transfer Request	Channel/Overflow Interrupt
0	0	The channel/overflow DMA transfer request is not generated.	The channel/overflow interrupt is not generated.
0	1	The channel/overflow DMA transfer request is not generated.	The channel/overflow interrupt is generated if (CHnF/TOF = 1).
1	0	The channel/overflow DMA transfer request is generated if (CHnF/TOF = 1).	The channel/overflow interrupt is not generated.
1	1	The channel/overflow DMA transfer request is generated if (CHnF/TOF = 1).	The channel/overflow interrupt is generated if (CHnF/TOF = 1).

If DMA = 1, the CHnF/TOF bit can be cleared either by DMA transfer done or writing a one to CHnF/TOF bit (see the following table).

**Table 29-7. Clear CHnF/TOF Bit**

DMA	How CHnF/TOF Bit Can Be Cleared
0	CHnF/TOF bit is cleared by writing a 1 to CHnF/TOF bit.
1	CHnF/TOF bit is cleared either when the DMA transfer is done or by writing a 1 to CHnF/TOF bit.

## 29.5.10 Output triggers

The TPM generates output triggers for the counter and each channel that can be used to trigger events in other peripherals.

The counter trigger asserts whenever the TOF is set and remains asserted until the next increment.

Each TPM channel generates both a pre-trigger output and a trigger output. The pre-trigger output asserts whenever the CHnF is set, the trigger output asserts on the first counter increment after the pre-trigger asserts, and then both the trigger and pre-trigger negate on the first counter increment after the trigger asserts.

## 29.5.11 Reset Overview

The TPM is reset whenever any chip reset occurs.

When the TPM exits from reset:

- the TPM counter and the prescaler counter are zero and are stopped (CMOD[1:0] = 0:0);
- the timer overflow interrupt is zero;
- the channels interrupts are zero;
- the channels are in input capture mode;
- the channels outputs are zero;
- the channels pins are not controlled by TPM (ELS(n)B:ELS(n)A = 0:0).

## 29.5.12 TPM Interrupts

This section describes TPM interrupts.

### 29.5.12.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

### 29.5.12.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

# Chapter 30

## Periodic Interrupt Timer (PIT)

### 30.1 Chip-specific PIT information

#### 30.1.1 PIT/DMA periodic trigger assignments

The PIT generates periodic trigger events to the DMA channel mux as shown in this table.

**Table 30-1. PIT channel assignments for periodic DMA triggering**

PIT channel	DMA channel number
PIT Channel 0	DMA Channel 0
PIT Channel 1	DMA Channel 1

#### 30.1.2 PIT/ADC triggers

PIT triggers are selected as ADCx trigger sources using the bits of SIM\_SOPT7[ADCxTRGSEL]. For more details, see the [SIM](#) chapter.

#### 30.1.3 PIT/TPM triggers

PIT triggers are selected as TPMx trigger sources using the TPMx\_CONF[TRGSEL] bits in the TPM module. For more details, refer to TPM chapter.

## 30.1.4 PIT/DAC triggers

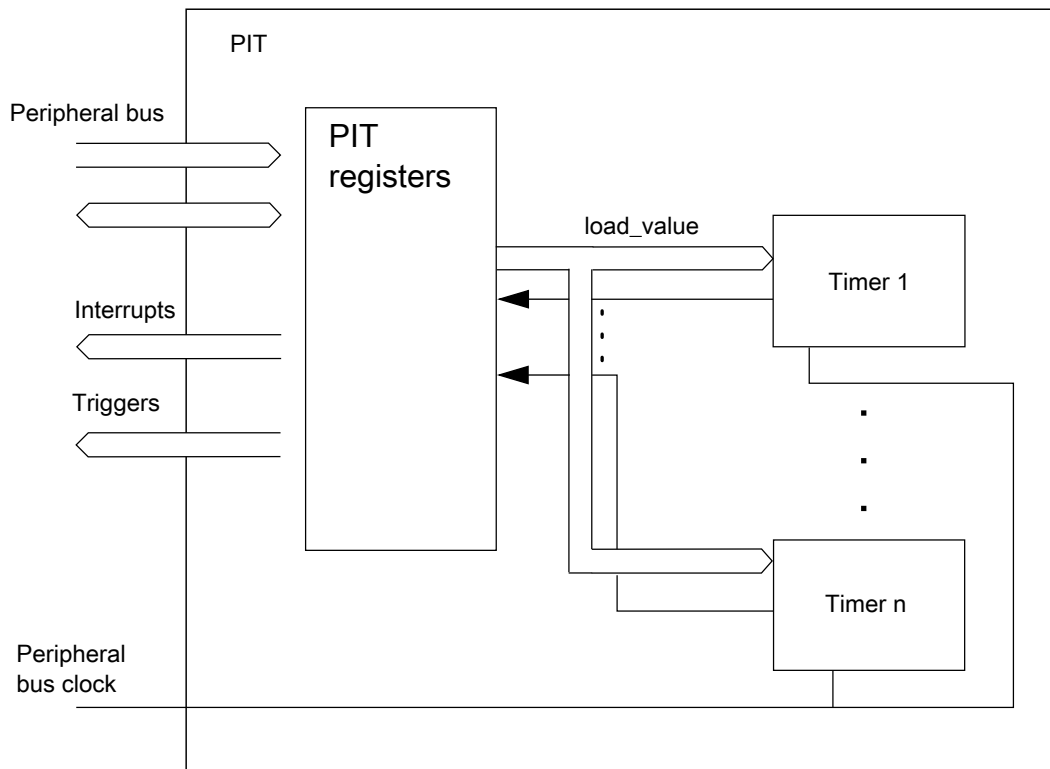
PIT Channel 0 is configured as the DAC hardware trigger source. For more details, see the [DAC](#) chapter.

## 30.2 Introduction

The PIT module is an array of timers that can be used to raise interrupts and trigger DMA channels.

### 30.2.1 Block diagram

The following figure shows the block diagram of the PIT module.



**Figure 30-1. Block diagram of the PIT**

### NOTE

See the chip-specific PIT information for the number of PIT channels used in this MCU.

## 30.2.2 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

## 30.3 Signal description

The PIT module has no external pins.

## 30.4 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**

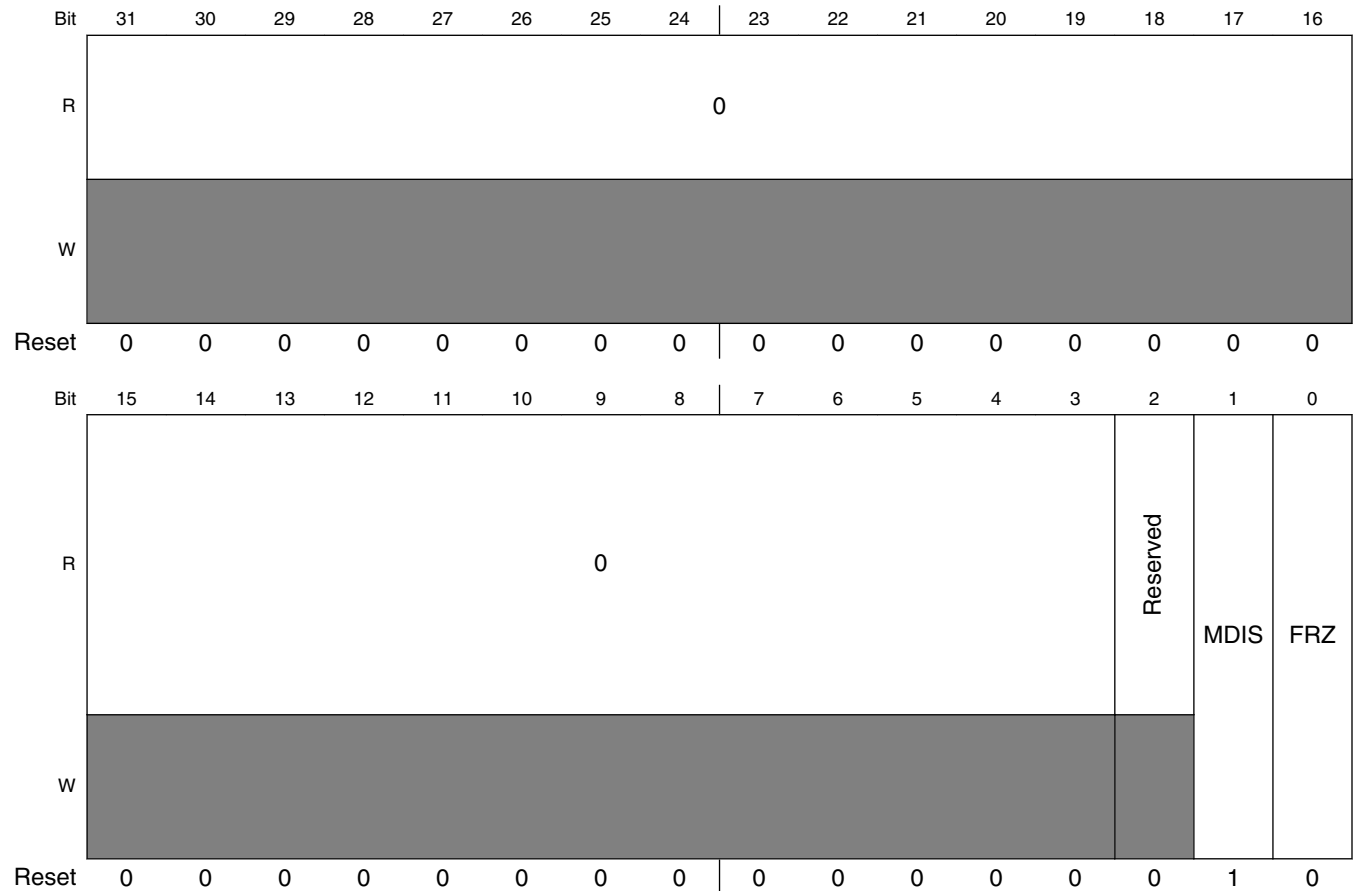
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_7000	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0002h	<a href="#">30.4.1/480</a>
4003_70E0	PIT Upper Lifetime Timer Register (PIT_LTMR64H)	32	R	0000_0000h	<a href="#">30.4.2/481</a>
4003_70E4	PIT Lower Lifetime Timer Register (PIT_LTMR64L)	32	R	0000_0000h	<a href="#">30.4.3/481</a>
4003_7100	Timer Load Value Register (PIT_LDVAL0)	32	R/W	0000_0000h	<a href="#">30.4.4/482</a>
4003_7104	Current Timer Value Register (PIT_CVAL0)	32	R	0000_0000h	<a href="#">30.4.5/482</a>
4003_7108	Timer Control Register (PIT_TCTRL0)	32	R/W	0000_0000h	<a href="#">30.4.6/483</a>
4003_710C	Timer Flag Register (PIT_TFLG0)	32	R/W	0000_0000h	<a href="#">30.4.7/484</a>
4003_7110	Timer Load Value Register (PIT_LDVAL1)	32	R/W	0000_0000h	<a href="#">30.4.4/482</a>
4003_7114	Current Timer Value Register (PIT_CVAL1)	32	R	0000_0000h	<a href="#">30.4.5/482</a>
4003_7118	Timer Control Register (PIT_TCTRL1)	32	R/W	0000_0000h	<a href="#">30.4.6/483</a>
4003_711C	Timer Flag Register (PIT_TFLG1)	32	R/W	0000_0000h	<a href="#">30.4.7/484</a>

### 30.4.1 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

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.

Table continues on the next page...



**PIT\_MCR field descriptions (continued)**

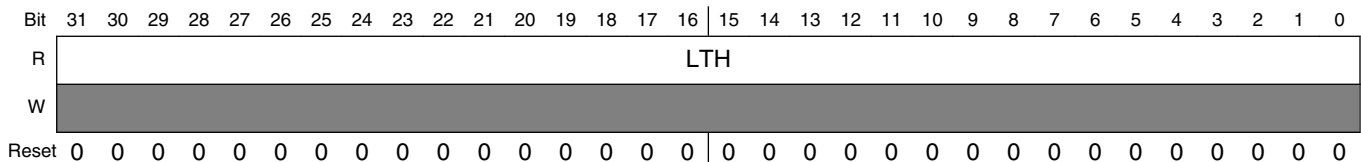
Field	Description
0 FRZ	Freeze  Allows the timers to be stopped when the device enters the Debug mode.  0 Timers continue to run in Debug mode. 1 Timers are stopped in Debug mode.

**30.4.2 PIT Upper Lifetime Timer Register (PIT\_LTMR64H)**

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

Access: User read only

Address: 4003\_7000h base + E0h offset = 4003\_70E0h

**PIT\_LTMR64H field descriptions**

Field	Description
LTH	Life Timer value  Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

**30.4.3 PIT Lower Lifetime Timer Register (PIT\_LTMR64L)**

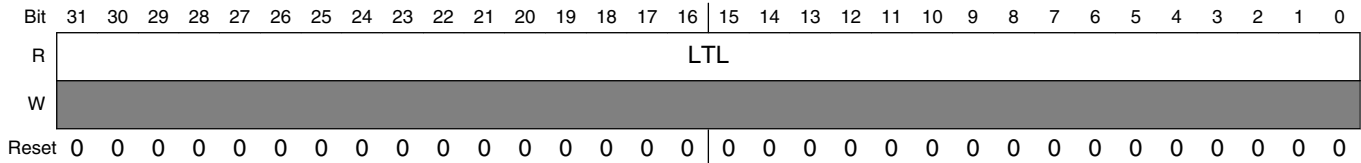
This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Access: User read only

## Memory map/register description

Address: 4003\_7000h base + E4h offset = 4003\_70E4h



### PIT\_LTMR64L field descriptions

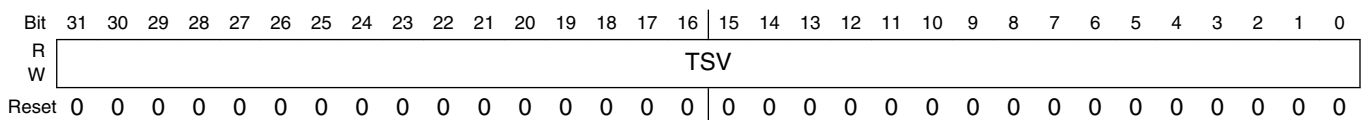
Field	Description
LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

## 30.4.4 Timer Load Value Register (PIT\_LDVALn)

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 1d



### PIT\_LDVALn field descriptions

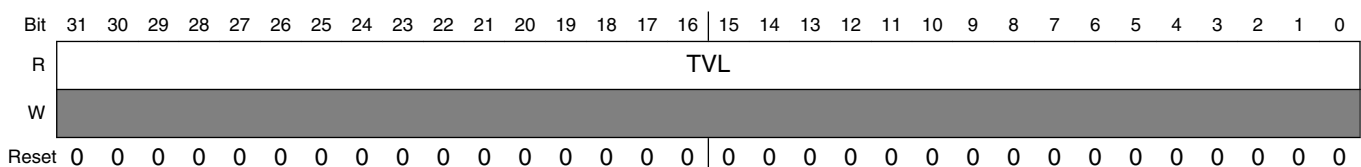
Field	Description
TSV	Timer Start Value Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

## 30.4.5 Current Timer Value Register (PIT\_CVALn)

These registers indicate the current timer position.

Access: User read only

Address: 4003\_7000h base + 104h offset + (16d × i), where i=0d to 1d



PIT\_CVAL<sub>n</sub> field descriptions

Field	Description
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>

30.4.6 Timer Control Register (PIT\_TCTRL<sub>n</sub>)

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 1d

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

PIT\_TCTRL<sub>n</sub> field descriptions

Field	Description
31–3 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
2 CHN	<p>Chain Mode</p> <p>When activated, Timer n-1 needs to expire before timer n can decrement by 1.</p> <p>Timer 0 cannot be chained.</p> <p>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.</p>
1 TIE	<p>Timer Interrupt Enable</p> <p>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.</p> <p>0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.</p>
0 TEN	<p>Timer Enable</p> <p>Enables or disables the timer.</p>

Table continues on the next page...

**PIT\_TCTRLn field descriptions (continued)**

Field	Description
0	Timer n is disabled.
1	Timer n is enabled.

**30.4.7 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 1d

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																
W	[Shaded]																
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	Timer Interrupt Flag  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.  0 Timeout has not yet occurred. 1 Timeout has occurred.

**30.5 Functional description**

This section provides the functional description of the module.

## 30.5.1 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.

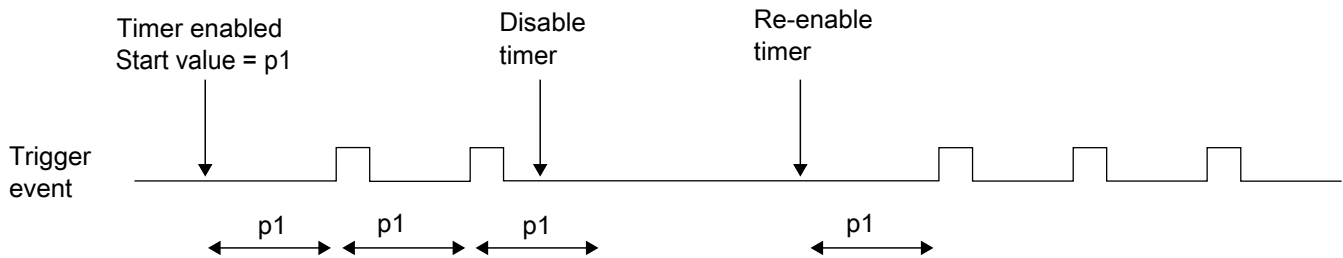
### 30.5.1.1 Timers

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.

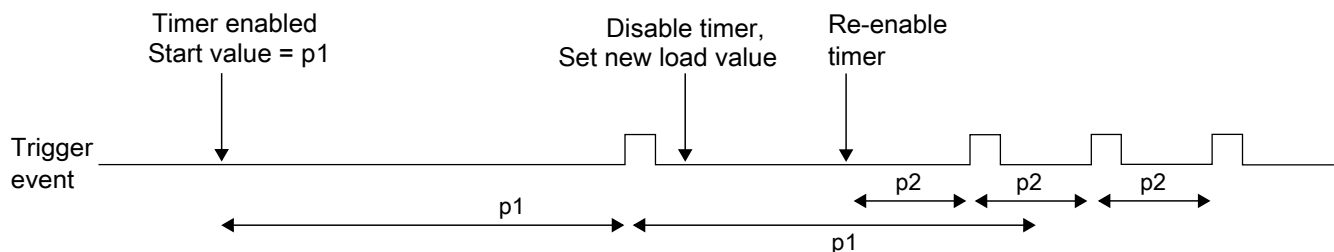
If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN]. See the following figure.



**Figure 30-2. 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 30-3. 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.

## Initialization and application information

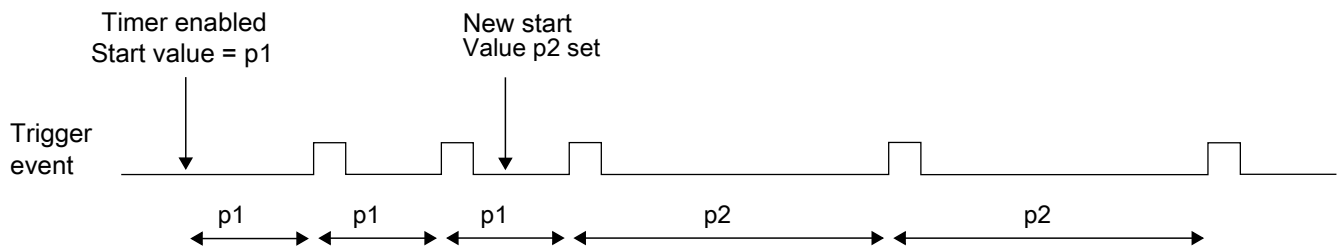


Figure 30-4. Dynamically setting a new load value

### 30.5.1.2 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.

### 30.5.2 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].

### 30.5.3 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.

## 30.6 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:

$\text{LDVAL trigger} = (\text{period} / \text{clock period}) - 1$

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
```

## 30.7 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].

## Example configuration for the lifetime timer

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
```

## 30.8 Example configuration for the lifetime timer

To configure the lifetime timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the CTRL register, then the LDVAL registers need to be set to the maximum value.

The timer is a downcounter.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;

// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 1 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1

// Timer 0
```



```
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime, read first LTMR64H and then LTMR64L.

```
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```



# Chapter 31

## Low-Power Timer (LPTMR)

### 31.1 Chip-specific LPTMR information

#### 31.1.1 LPTMR instantiation information

The low-power timer (LPTMR) allows operation during all power modes. The LPTMR can operate as a real-time interrupt or pulse accumulator. It includes a  $2^N$  prescaler (real-time interrupt mode) or glitch filter (pulse accumulator mode).

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

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

#### 31.1.2 LPTMR pulse counter input options

LPTMR\_CSR[TPS] configures the input source used in pulse counter mode. The following table shows the chip-specific input assignments for this field.

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

### 31.1.3 LPTMR prescaler/glitch filter clocking options

The prescaler and glitch filter of the LPTMR module can be clocked from one of four sources determined by LPTMR0\_PSR[PCS]. The following table shows the chip-specific clock assignments for this field.

#### 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 LLS and VLLS modes)
01	1	LPO—1 kHz clock (not available in VLLS0 mode)
10	2	ERCLK32K (not available in VLLS0 mode when using 32 kHz oscillator)
11	3	OSCECLK—external reference clock (not available in VLLS0 mode)

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

## 31.2 Introduction

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.

### 31.2.1 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

## 31.2.2 Modes of operation

The following table describes the operation of the LPTMR module in various modes.

**Table 31-1. 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.

## 31.3 LPTMR signal descriptions

**Table 31-2. LPTMR signal descriptions**

Signal	I/O	Description
LPTMR0_ALT $n$	I	Pulse Counter Input pin

### 31.3.1 Detailed signal descriptions

**Table 31-3. 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.

## 31.4 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="#">31.4.1/494</a>
4004_0004	Low Power Timer Prescale Register (LPTMR0_PSR)	32	R/W	0000_0000h	<a href="#">31.4.2/495</a>
4004_0008	Low Power Timer Compare Register (LPTMR0_CMR)	32	R/W	0000_0000h	<a href="#">31.4.3/497</a>
4004_000C	Low Power Timer Counter Register (LPTMR0_CNR)	32	R/W	0000_0000h	<a href="#">31.4.4/497</a>

### 31.4.1 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	[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								TCF	TIE	TPS	TPP	TFC	TMS	TEN	
W	[Shaded]								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

Table continues on the next page...

## LPTMRx\_CSR field descriptions (continued)

Field	Description
	<p>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 for information on the connections to these inputs.</p> <p>00 Pulse counter input 0 is selected.  01 Pulse counter input 1 is selected.  10 Pulse counter input 2 is selected.  11 Pulse counter input 3 is selected.</p>
3 TPP	<p>Timer Pin Polarity</p> <p>Configures the polarity of the input source in Pulse Counter mode. TPP must be changed only when the LPTMR is disabled.</p> <p>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.</p>
2 TFC	<p>Timer Free-Running Counter</p> <p>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.</p> <p>0 CNR is reset whenever TCF is set.  1 CNR is reset on overflow.</p>
1 TMS	<p>Timer Mode Select</p> <p>Configures the mode of the LPTMR. TMS must be altered only when the LPTMR is disabled.</p> <p>0 Time Counter mode.  1 Pulse Counter mode.</p>
0 TEN	<p>Timer Enable</p> <p>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.</p> <p>0 LPTMR is disabled and internal logic is reset.  1 LPTMR is enabled.</p>

## 31.4.2 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>Prescaler 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>
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.

## 31.4.3 Low Power Timer Compare Register (LPTMRx\_CMCR)

Address: 4004\_0000h base + 8h offset = 4004\_0008h

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

## 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.
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.

## 31.4.4 Low Power Timer Counter Register (LPTMRx\_CNCR)

Address: 4004\_0000h base + Ch offset = 4004\_000Ch

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

## LPTMRx\_CNCR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNTER	Counter Value  The CNR returns the value of the LPTMR counter at the time this register was last written. Writing the CNR will latch the current value of the LPTMR for subsequent reading, the value written is ignored.

## 31.5 Functional description

### 31.5.1 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.

### 31.5.2 LPTMR clocking

The LPTMR prescaler/glitch filter can be clocked by one of the four clocks. The clock source must be enabled before the LPTMR is enabled.

#### NOTE

The clock source selected need to be configured to remain enabled in low-power modes, otherwise the LPTMR will not operate during low-power modes.

In Pulse Counter mode with the prescaler/glitch filter bypassed, the selected input source directly clocks the CNR and no other clock source is required. To minimize power in this case, configure the prescaler clock source for a clock that is not toggling.

#### NOTE

The clock source or pulse input source selected for the LPTMR should not exceed the frequency  $f_{LPTMR}$  defined in the device datasheet.

### 31.5.3 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.

**31.5.3.1 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.

**31.5.3.2 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.

**31.5.3.3 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.

### 31.5.3.4 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.

### 31.5.4 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.

### 31.5.5 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.

### 31.5.6 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.

### 31.5.7 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 32

## Real Time Clock (RTC)

### 32.1 Chip-specific RTC information

#### 32.1.1 RTC Instantiation Information

RTC prescaler is clocked by ERCLK32K.

RTC is reset on POR Only.

RTC\_CR[OSCE] can override the configuration of the System OSC, configuring the OSC for 32 kHz crystal operation in all power modes except VLLS0, and through any System Reset. When OSCE is enabled, the RTC also overrides the capacitor configurations. So make sure to use low-range crystal (30KHz - 40KHz) when using RTC\_CR to configure the OSC.

#### 32.1.2 RTC\_CLKOUT options

RTC\_CLKOUT pin can be driven either with the RTC 1 Hz output or with the OSCERCLK on-chip clock source. Control for this option is through SIM\_SOPT2[RTCCLKOUTSEL].

When SIM\_SOPT2[RTCCLKOUTSEL] = 0, the RTC 1 Hz clock is output on the RTC\_CLKOUT pin. When SIM\_SOPT2[RTCCLKOUTSEL] = 1, OSCERCLK clock is output on the RTC\_CLKOUT pin.

### 32.2 Introduction

## 32.2.1 Features

The RTC module features include:

- 32-bit seconds counter with roll-over protection and 32-bit alarm
- 16-bit prescaler with compensation that can correct errors between 0.12 ppm and 3906 ppm
- Register write protection
  - Lock register requires POR or software reset to enable write access
- 1 Hz square wave output with optional interrupt

## 32.2.2 Modes of operation

The RTC remains functional in all low power modes and can generate an interrupt to exit any low power mode.

## 32.2.3 RTC signal descriptions

Table 32-1. RTC signal descriptions

Signal	Description	I/O
RTC_CLKOUT	1 Hz square-wave output or OSCERCLK	O

### 32.2.3.1 RTC clock output

The clock to the seconds counter is available on the RTC\_CLKOUT signal. It is a 1 Hz square wave output. See [RTC\\_CLKOUT options](#) for details.

## 32.3 Register definition

All registers must be accessed using 32-bit writes and all register accesses incur three wait states.

Write accesses to any register by non-supervisor mode software, when the supervisor access bit in the control register is clear, will terminate with a bus error.

Read accesses by non-supervisor mode software complete as normal.



Writing to a register protected by the lock register does not generate a bus error, but the write will not complete.

### RTC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_D000	RTC Time Seconds Register (RTC_TSR)	32	R/W	0000_0000h	<a href="#">32.3.1/505</a>
4003_D004	RTC Time Prescaler Register (RTC_TPR)	32	R/W	0000_0000h	<a href="#">32.3.2/505</a>
4003_D008	RTC Time Alarm Register (RTC_TAR)	32	R/W	0000_0000h	<a href="#">32.3.3/506</a>
4003_D00C	RTC Time Compensation Register (RTC_TCR)	32	R/W	0000_0000h	<a href="#">32.3.4/506</a>
4003_D010	RTC Control Register (RTC_CR)	32	R/W	0000_0000h	<a href="#">32.3.5/508</a>
4003_D014	RTC Status Register (RTC_SR)	32	R/W	0000_0001h	<a href="#">32.3.6/510</a>
4003_D018	RTC Lock Register (RTC_LR)	32	R/W	0000_00FFh	<a href="#">32.3.7/511</a>
4003_D01C	RTC Interrupt Enable Register (RTC_IER)	32	R/W	0000_0007h	<a href="#">32.3.8/512</a>

### 32.3.1 RTC Time Seconds Register (RTC\_TSR)

Address: 4003\_D000h base + 0h offset = 4003\_D000h

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	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

#### RTC\_TSR field descriptions

Field	Description
TSR	Time Seconds Register  When the time counter is enabled, the TSR is read only and increments once a second provided SR[TOF] or SR[TIF] are not set. The time counter will read as zero when SR[TOF] or SR[TIF] are set. When the time counter is disabled, the TSR can be read or written. Writing to the TSR when the time counter is disabled will clear the SR[TOF] and/or the SR[TIF]. Writing to TSR with zero is supported, but not recommended because TSR will read as zero when SR[TIF] or SR[TOF] are set (indicating the time is invalid).

### 32.3.2 RTC Time Prescaler Register (RTC\_TPR)

Address: 4003\_D000h base + 4h offset = 4003\_D004h

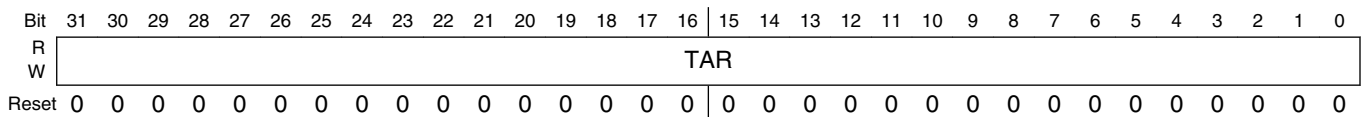
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																0															
W	0																0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### RTC\_TPR field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TPR	Time Prescaler Register  When the time counter is enabled, the TPR is read only and increments every 32.768 kHz clock cycle. The time counter will read as zero when SR[TOF] or SR[TIF] are set. When the time counter is disabled, the TPR can be read or written. The TSR[TSR] increments when bit 14 of the TPR transitions from a logic one to a logic zero.

### 32.3.3 RTC Time Alarm Register (RTC\_TAR)

Address: 4003\_D000h base + 8h offset = 4003\_D008h

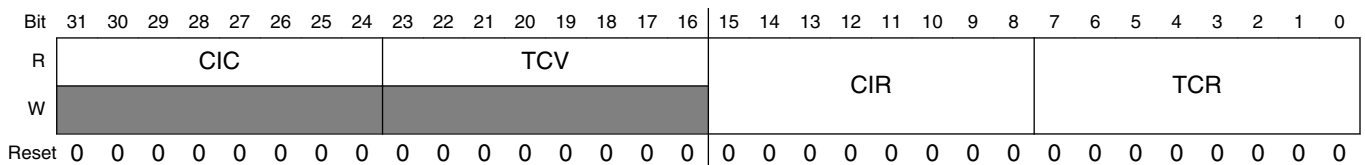


### RTC\_TAR field descriptions

Field	Description
TAR	Time Alarm Register  When the time counter is enabled, the SR[TAF] is set whenever the TAR[TAR] equals the TSR[TSR] and the TSR[TSR] increments. Writing to the TAR clears the SR[TAF].

### 32.3.4 RTC Time Compensation Register (RTC\_TCR)

Address: 4003\_D000h base + Ch offset = 4003\_D00Ch



### RTC\_TCR field descriptions

Field	Description
31–24 CIC	Compensation Interval Counter  Current value of the compensation interval counter. If the compensation interval counter equals zero then it is loaded with the contents of the CIR. If the CIC does not equal zero then it is decremented once a second.
23–16 TCV	Time Compensation Value

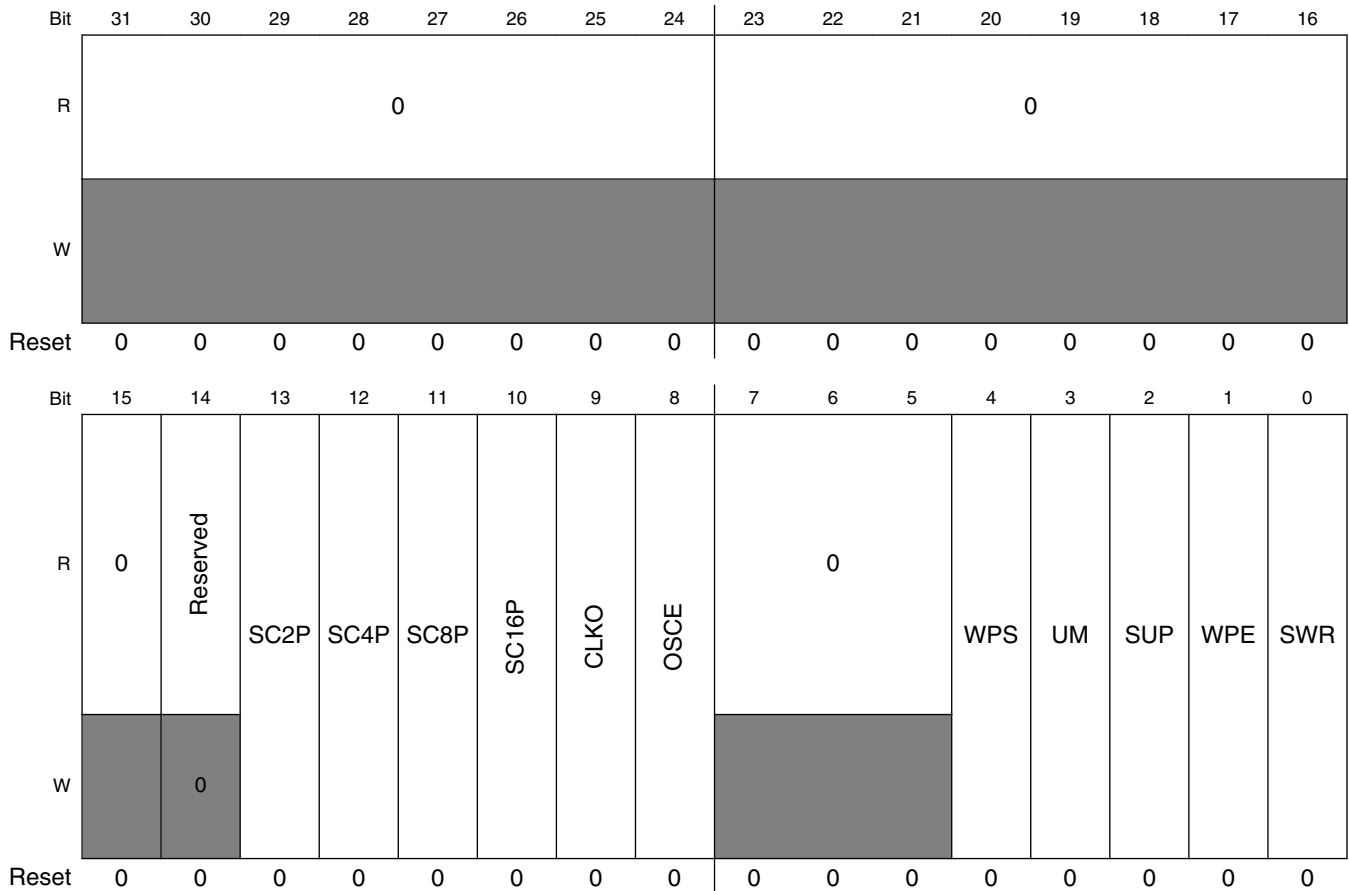
Table continues on the next page...

## RTC\_TCR field descriptions (continued)

Field	Description
	Current value used by the compensation logic for the present second interval. Updated once a second if the CIC equals 0 with the contents of the TCR field. If the CIC does not equal zero then it is loaded with zero (compensation is not enabled for that second increment).
15–8 CIR	<p>Compensation Interval Register</p> <p>Configures the compensation interval in seconds from 1 to 256 to control how frequently the TCR should adjust the number of 32.768 kHz cycles in each second. The value written should be one less than the number of seconds. For example, write zero to configure for a compensation interval of one second. This register is double buffered and writes do not take affect until the end of the current compensation interval.</p>
TCR	<p>Time Compensation Register</p> <p>Configures the number of 32.768 kHz clock cycles in each second. This register is double buffered and writes do not take affect until the end of the current compensation interval.</p> <p>80h Time Prescaler Register overflows every 32896 clock cycles.  ... ..  FFh Time Prescaler Register overflows every 32769 clock cycles.  00h Time Prescaler Register overflows every 32768 clock cycles.  01h Time Prescaler Register overflows every 32767 clock cycles.  .... ..  7Fh Time Prescaler Register overflows every 32641 clock cycles.</p>

### 32.3.5 RTC Control Register (RTC\_CR)

Address: 4003\_D000h base + 10h offset = 4003\_D010h



**RTC\_CR field descriptions**

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 Reserved	This field is reserved. It must always be written to 0.
13 SC2P	Oscillator 2pF Load Configure 0 Disable the load. 1 Enable the additional load.
12 SC4P	Oscillator 4pF Load Configure 0 Disable the load. 1 Enable the additional load.

*Table continues on the next page...*

## RTC\_CR field descriptions (continued)

Field	Description
11 SC8P	Oscillator 8pF Load Configure 0 Disable the load. 1 Enable the additional load.
10 SC16P	Oscillator 16pF Load Configure 0 Disable the load. 1 Enable the additional load.
9 CLKO	Clock Output 0 The 32 kHz clock is output to other peripherals. 1 The 32 kHz clock is not output to other peripherals.
8 OSCE	Oscillator Enable 0 32.768 kHz oscillator is disabled. 1 32.768 kHz oscillator is enabled. After setting this bit, wait the oscillator startup time before enabling the time counter to allow the 32.768 kHz clock time to stabilize.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 WPS	Wakeup Pin Select The wakeup pin is optional and not available on all devices. 0 Wakeup pin asserts (active low, open drain) if the RTC interrupt asserts or the wakeup pin is turned on. 1 Wakeup pin instead outputs the RTC 32kHz clock, provided the wakeup pin is turned on and the 32kHz clock is output to other peripherals.
3 UM	Update Mode Allows SR[TCE] to be written even when the Status Register is locked. When set, the SR[TCE] can always be written if the SR[TIF] or SR[TOF] are set or if the SR[TCE] is clear. 0 Registers cannot be written when locked. 1 Registers can be written when locked under limited conditions.
2 SUP	Supervisor Access 0 Non-supervisor mode write accesses are not supported and generate a bus error. 1 Non-supervisor mode write accesses are supported.
1 WPE	Wakeup Pin Enable The wakeup pin is optional and not available on all devices. 0 Wakeup pin is disabled. 1 Wakeup pin is enabled and wakeup pin asserts if the RTC interrupt asserts or the wakeup pin is turned on.
0 SWR	Software Reset 0 No effect. 1 Resets all RTC registers except for the SWR bit . The SWR bit is cleared by POR and by software explicitly clearing it.

### 32.3.6 RTC Status Register (RTC\_SR)

Address: 4003\_D000h base + 14h offset = 4003\_D014h

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								TCE	0	TAF	TOF	TIF				
W	[Shaded]								[Shaded]	[Shaded]	[Shaded]	[Shaded]					
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	1

#### RTC\_SR field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 TCE	Time Counter Enable  When time counter is disabled the TSR register and TPR register are writeable, but do not increment. When time counter is enabled the TSR register and TPR register are not writeable, but increment.  0 Time counter is disabled. 1 Time counter is enabled.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 TAF	Time Alarm Flag  Time alarm flag is set when the TAR[TAR] equals the TSR[TSR] and the TSR[TSR] increments. This bit is cleared by writing the TAR register.  0 Time alarm has not occurred. 1 Time alarm has occurred.
1 TOF	Time Overflow Flag  Time overflow flag is set when the time counter is enabled and overflows. The TSR and TPR do not increment and read as zero when this bit is set. This bit is cleared by writing the TSR register when the time counter is disabled.  0 Time overflow has not occurred. 1 Time overflow has occurred and time counter is read as zero.
0 TIF	Time Invalid Flag  The time invalid flag is set on POR or software reset. The TSR and TPR do not increment and read as zero when this bit is set. This bit is cleared by writing the TSR register when the time counter is disabled.  0 Time is valid. 1 Time is invalid and time counter is read as zero.

### 32.3.7 RTC Lock Register (RTC\_LR)

Address: 4003\_D000h base + 18h offset = 4003\_D018h

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								1	LRL	SRL	CRL	TCL	1			
W																	
Reset	0	0	0	0	0	0	0	0		1	1	1	1	1	1	1	1

#### RTC\_LR field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6 LRL	Lock Register Lock After being cleared, this bit can be set only by POR or software reset. 0 Lock Register is locked and writes are ignored. 1 Lock Register is not locked and writes complete as normal.
5 SRL	Status Register Lock After being cleared, this bit can be set only by POR or software reset. 0 Status Register is locked and writes are ignored. 1 Status Register is not locked and writes complete as normal.
4 CRL	Control Register Lock After being cleared, this bit can only be set by POR. 0 Control Register is locked and writes are ignored. 1 Control Register is not locked and writes complete as normal.
3 TCL	Time Compensation Lock After being cleared, this bit can be set only by POR or software reset. 0 Time Compensation Register is locked and writes are ignored. 1 Time Compensation Register is not locked and writes complete as normal.
Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

### 32.3.8 RTC Interrupt Enable Register (RTC\_IER)

Address: 4003\_D000h base + 1Ch offset = 4003\_D01Ch

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								WPON	Reserved	TSIE	Reserved	TAIE	TOIE	TIIE	
W	[Shaded]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

**RTC\_IER 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 WPON	Wakeup Pin On  The wakeup pin is optional and not available on all devices. Whenever the wakeup pin is enabled and this bit is set, the wakeup pin will assert.  0 No effect. 1 If the wakeup pin is enabled, then the wakeup pin will assert.
6–5 Reserved	This field is reserved.
4 TSIE	Time Seconds Interrupt Enable  The seconds interrupt is an edge-sensitive interrupt with a dedicated interrupt vector. It is generated once a second and requires no software overhead (there is no corresponding status flag to clear).  0 Seconds interrupt is disabled. 1 Seconds interrupt is enabled.
3 Reserved	This field is reserved.
2 TAIE	Time Alarm Interrupt Enable  0 Time alarm flag does not generate an interrupt. 1 Time alarm flag does generate an interrupt.
1 TOIE	Time Overflow Interrupt Enable

Table continues on the next page...



**RTC\_IER field descriptions (continued)**

Field	Description
	0 Time overflow flag does not generate an interrupt. 1 Time overflow flag does generate an interrupt.
0 TIIE	Time Invalid Interrupt Enable 0 Time invalid flag does not generate an interrupt. 1 Time invalid flag does generate an interrupt.

## 32.4 Functional description

### 32.4.1 Power, clocking, and reset

The RTC is an always powered block that remains active in all low power modes.

The time counter within the RTC is clocked by a 32.768 kHz clock sourced from an external crystal using the oscillator.

The power-on-reset signal initializes all RTC registers to their default state. A software reset bit can also initialize all RTC registers.

#### 32.4.1.1 Oscillator control

The 32.768 kHz crystal oscillator is disabled at POR and must be enabled by software. After enabling the crystal oscillator, wait the oscillator startup time before setting SR[TCE] or using the oscillator clock external to the RTC.

The crystal oscillator includes tunable capacitors that can be configured by software. Do not change the capacitance unless the oscillator is disabled.

#### 32.4.1.2 Software reset

Writing 1 to CR[SWR] forces the equivalent of a POR to the rest of the RTC module. CR[SWR] is not affected by the software reset and must be cleared by software.

### 32.4.1.3 Supervisor access

When the supervisor access control bit is clear, only supervisor mode software can write to the RTC registers, non-supervisor mode software will generate a bus error. Both supervisor and non-supervisor mode software can always read the RTC registers.

### 32.4.2 Time counter

The time counter consists of a 32-bit seconds counter that increments once every second and a 16-bit prescaler register that increments once every 32.768 kHz clock cycle.

Reading the time counter (either seconds or prescaler) while it is incrementing may return invalid data due to synchronization of the read data bus. If it is necessary for software to read the prescaler or seconds counter when they could be incrementing, it is recommended that two read accesses are performed and that software verifies that the same data was returned for both reads.

The time seconds register and time prescaler register can be written only when SR[TCE] is clear. Always write to the prescaler register before writing to the seconds register, because the seconds register increments on the falling edge of bit 14 of the prescaler register.

The time prescaler register increments provided SR[TCE] is set, SR[TIF] is clear, SR[TOF] is clear, and the 32.768 kHz clock source is present. After enabling the oscillator, wait the oscillator startup time before setting SR[TCE] to allow time for the oscillator clock output to stabilize.

If the time seconds register overflows then the SR[TOF] will set and the time prescaler register will stop incrementing. Clear SR[TOF] by initializing the time seconds register. The time seconds register and time prescaler register read as zero whenever SR[TOF] is set.

SR[TIF] is set on POR and software reset and is cleared by initializing the time seconds register. The time seconds register and time prescaler register read as zero whenever SR[TIF] is set.

### 32.4.3 Compensation

The compensation logic provides an accurate and wide compensation range and can correct errors as high as 3906 ppm and as low as 0.12 ppm. The compensation factor must be calculated externally to the RTC and supplied by software to the compensation

register. The RTC itself does not calculate the amount of compensation that is required, although the 1 Hz clock is output to an external pin in support of external calibration logic.

Crystal compensation can be supported by using firmware and crystal characteristics to determine the compensation amount. Temperature compensation can be supported by firmware that periodically measures the external temperature via ADC and updates the compensation register based on a look-up table that specifies the change in crystal frequency over temperature.

The compensation logic alters the number of 32.768 kHz clock cycles it takes for the prescaler register to overflow and increment the time seconds counter. The time compensation value is used to adjust the number of clock cycles between -127 and +128. Cycles are added or subtracted from the prescaler register when the prescaler register equals 0x3FFF and then increments. The compensation interval is used to adjust the frequency at which the time compensation value is used, that is, from once a second to once every 256 seconds.

Updates to the time compensation register will not take effect until the next time the time seconds register increments and provided the previous compensation interval has expired. When the compensation interval is set to other than once a second then the compensation is applied in the first second interval and the remaining second intervals receive no compensation.

Compensation is disabled by configuring the time compensation register to zero.

#### 32.4.4 Time alarm

The Time Alarm register (TAR), SR[TAF], and IER[TAIE] allow the RTC to generate an interrupt at a predefined time. The 32-bit TAR is compared with the 32-bit Time Seconds register (TSR) each time it increments. SR[TAF] will set when TAR equals TSR and TSR increments.

SR[TAF] is cleared by writing TAR. This will usually be the next alarm value, although writing a value that is less than TSR, such as 0, will prevent SR[TAF] from setting again. SR[TAF] cannot otherwise be disabled, although the interrupt it generates is enabled or disabled by IER[TAIE].

### 32.4.5 Update mode

The Update Mode field in the Control register (CR[UM]) configures software write access to the Time Counter Enable (SR[TCE]) field. When CR[UM] is clear, SR[TCE] can be written only when LR[SRL] is set. When CR[UM] is set, SR[TCE] can also be written when SR[TCE] is clear or when SR[TIF] or SR[TOF] are set. This allows the time seconds and prescaler registers to be initialized whenever time is invalidated, while preventing the time seconds and prescaler registers from being changed on the fly. When LR[SRL] is set, CR[UM] has no effect on SR[TCE].

### 32.4.6 Register lock

The Lock register (LR) can be used to block write accesses to certain registers until the next POR or software reset. Locking the Control register (CR) will disable the software reset. Locking LR will block future updates to LR.

Write accesses to a locked register are ignored and do not generate a bus error.

### 32.4.7 Interrupt

The RTC interrupt is asserted whenever a status flag and the corresponding interrupt enable bit are both set. It is always asserted on POR, and software reset. The RTC interrupt is enabled at the chip level by enabling the chip-specific RTC clock gate control bit. The RTC interrupt can be used to wakeup the chip from any low-power mode.

The optional RTC seconds interrupt is an edge-sensitive interrupt with a dedicated interrupt vector that is generated once a second and requires no software overhead (there is no corresponding status flag to clear). It is enabled in the RTC by the time seconds interrupt enable bit and enabled at the chip level by setting the chip-specific RTC clock gate control bit. This interrupt is optional and may not be implemented on all devices.

# Chapter 33

## Serial Peripheral Interface (SPI)

### 33.1 Chip-specific SPI information

This device contains two SPI modules that support 16-bit data length. SPI1 includes a 4-deep FIFO. SPI0 is clocked on the bus clock. SPI1 is clocked from the system clock. SPI1 is therefore disabled in "Partial Stop Mode". The SPI supports DMA request and can operate in VLPS mode. When the SPI is operating in VLPS mode, it operates as a slave. SPI can wake the MCU from VLPS mode upon reception of SPI data in slave mode. SPI0 operates at maximum configurable speed — 12MHz in Master Mode (Bus/2). SPI1 operates at maximum configurable speed — 24MHz in Master Mode (System clock/2).

The following registers are not available in this device:

**Table 33-1. SPI register**

Absolute address	Register	Instance
0x4007_600A	SPI clear interrupt register (SPI0_CI)	SPI0
0x4007_600B	SPI control register 3 (SPI0_C3)	SPI0

#### NOTE

SPI0 has no SPI0\_CI and SPI0\_C3 and relative register bit in SPI0\_S

### 33.2 Introduction

The serial peripheral interface (SPI) module provides for full-duplex, synchronous, serial communication between the MCU and peripheral devices. These peripheral devices can include other microcontrollers, analog-to-digital converters, shift registers, sensors, and memories, among others.

The SPI runs at a baud rate up to the SPI module clock divided by two in master mode and up to the SPI module clock divided by four in slave mode. Software can poll the status flags, or SPI operation can be interrupt driven.

### NOTE

For the actual maximum SPI baud rate, refer to the Chip Configuration details and to the device's Data Sheet.

The SPI also supports a data length of 8 or 16 bits and includes a hardware match feature for the receive data buffer.

The SPI includes an internal DMA interface to support continuous SPI transmission through an on-chip DMA controller instead of through the CPU. This feature decreases CPU loading, allowing CPU time to be used for other work.

## 33.2.1 Features

The SPI includes these distinctive features:

- Master mode or slave mode operation
- Full-duplex or single-wire bidirectional mode
- Programmable transmit bit rate
- Double-buffered transmit and receive data register
- Serial clock phase and polarity options
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Control of SPI operation during wait mode
- Selectable MSB-first or LSB-first shifting
- Programmable 8- or 16-bit data transmission length
- Receive data buffer hardware match feature
- 64-bit FIFO mode for high speed/large amounts of data transfers
- Support transmission of both Transmit and Receive by DMA

## 33.2.2 Modes of operation

The SPI functions in the following three modes.

- Run mode

This is the basic mode of operation.

- Wait mode

SPI operation in Wait mode is a configurable low power mode, controlled by the SPISWAI bit located in the SPIx\_C2 register. In Wait mode, if C2[SPISWAI] is clear, the SPI operates like in Run mode. If C2[SPISWAI] is set, the SPI goes into a power conservative state, with the SPI clock generation turned off. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU enters Run mode. If the SPI is configured as a slave, reception and transmission of a byte continues, so that the slave stays synchronized to the master.

- Stop mode

To reduce power consumption, the SPI is inactive in stop modes where the peripheral bus clock is stopped but internal logic states are retained. If the SPI is configured as a master, any transmission in progress stops, but is resumed after the CPU enters run mode. If the SPI is configured as a slave, reception and transmission of a data continues, so that the slave stays synchronized to the master.

The SPI is completely disabled in Stop modes where the peripheral bus clock is stopped and internal logic states are not retained. When the CPU wakes from these Stop modes, all SPI register content is reset.

Detailed descriptions of operating modes appear in [Low-power mode options](#).

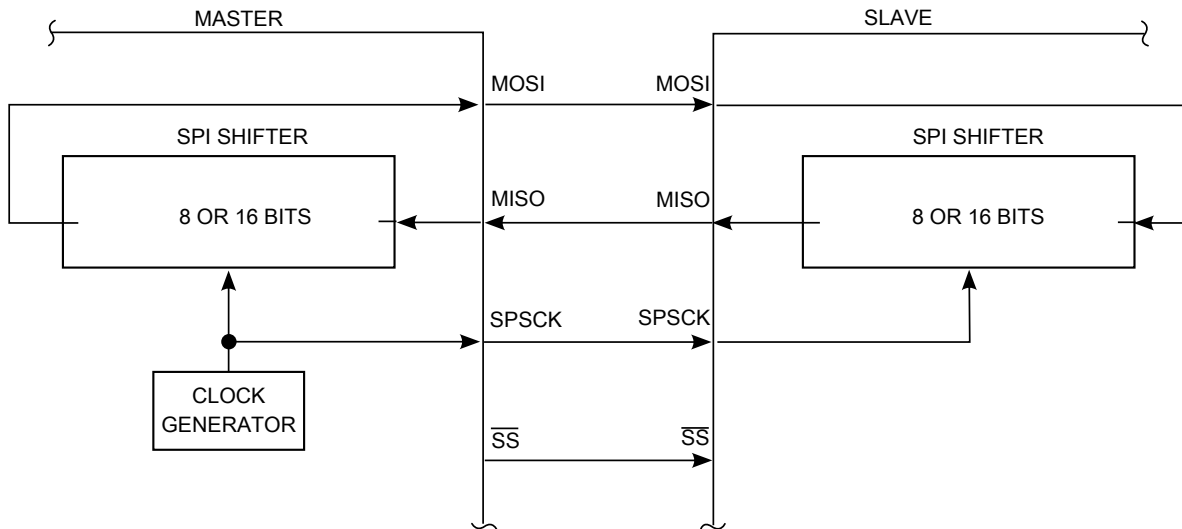
## 33.2.3 Block diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

### 33.2.3.1 SPI system block diagram

The following figure shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data

in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input (SS pin). In this system, the master device has configured its SS pin as an optional slave select output.



**Figure 33-1. SPI system connections**

### 33.2.3.2 SPI module block diagram

The following is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPIx\_DH:SPIx\_DL) and gets transferred to the SPI Shift Register at the start of a data transfer. After shifting in 8 bits or 16 bits (as determined by the SPIMODE bit) of data, the data is transferred into the double-buffered receiver where it can be read from SPIx\_DH:SPIx\_DL. Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the FIFO feature is supported: Additionally there is an 8-byte receive FIFO and an 8-byte transmit FIFO that (once enabled) provide features to allow fewer CPU interrupts to occur when transmitting/receiving high volume/high speed data. When FIFO mode is enabled, the SPI can still function in either 8-bit or 16-bit mode (as per SPIMODE bit) and three additional flags help monitor the FIFO status. Two of these flags can provide CPU interrupts.

When the SPI is configured as a master, the clock output is routed to the SPSCCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.



When the SPI is configured as a slave, the SPCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

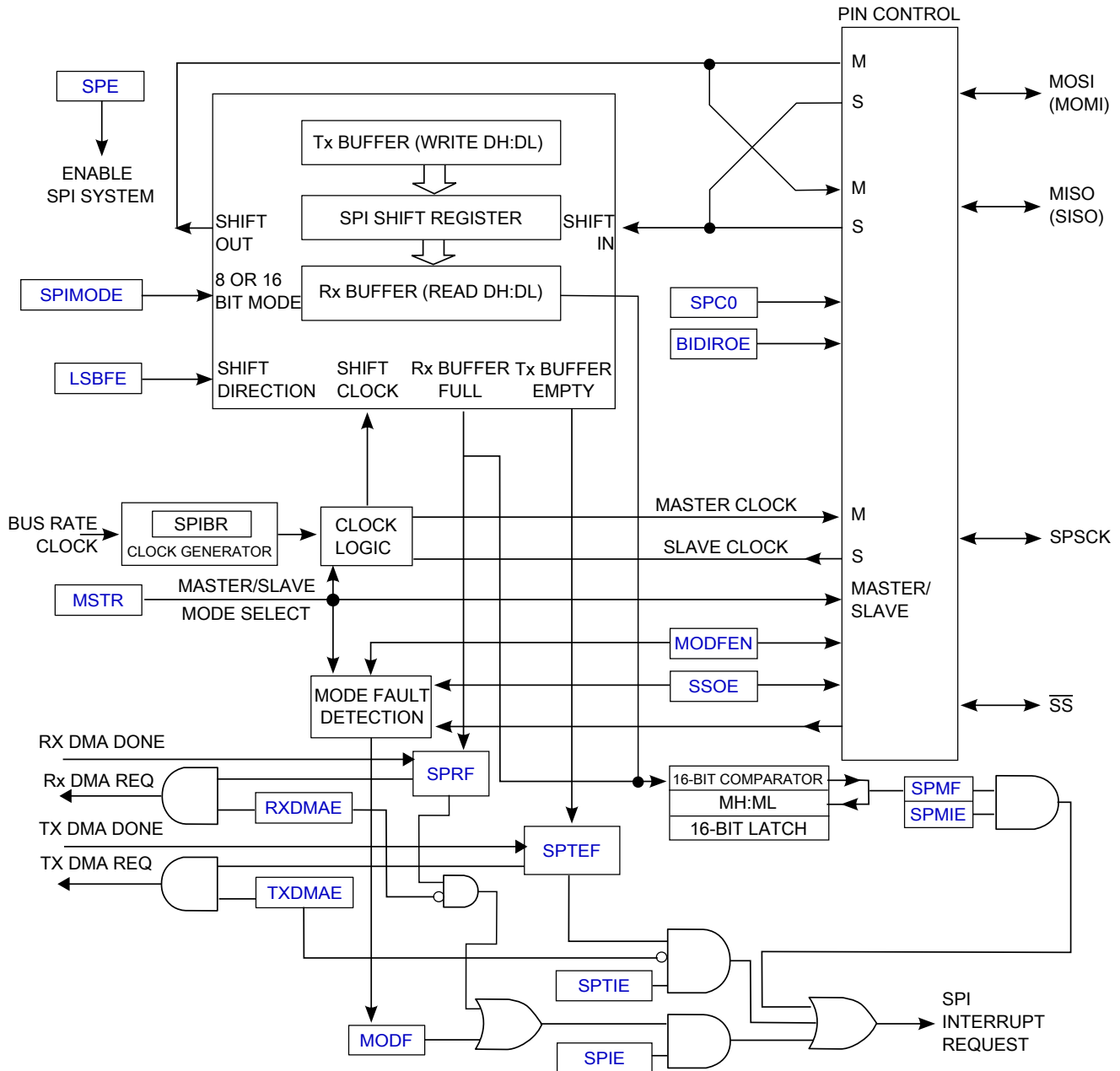


Figure 33-2. SPI module block diagram without FIFO



### 33.3.1 SPCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

### 33.3.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 is 0, this pin is the serial data input. If SPC0 is 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE is 0) or an output (BIDIROE is 1). If SPC0 is 1 and slave mode is selected, this pin is not used by the SPI and reverts to other functions (based on chip configuration).

### 33.3.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 is 0, this pin is the serial data output. If SPC0 is 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO), and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE is 0) or an output (BIDIROE is 1). If SPC0 is 1 and master mode is selected, this pin is not used by the SPI and reverts to other functions (based on chip configuration).

### 33.3.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN is 0), this pin is not used by the SPI and reverts to other functions (based on chip configuration). When the SPI is enabled as a master and MODFEN is 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE is 0) or as the slave select output (SSOE is 1).

## 33.4 Memory map/register definition

The SPI has 8-bit registers to select SPI options, to control baud rate, to report SPI status, to hold an SPI data match value, and for transmit/receive data.

**SPI memory map**

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4007_6000	SPI Status Register (SPI0_S)	8	R	20h	<a href="#">33.4.1/524</a>
4007_6001	SPI Baud Rate Register (SPI0_BR)	8	R/W	00h	<a href="#">33.4.2/528</a>
4007_6002	SPI Control Register 2 (SPI0_C2)	8	R/W	00h	<a href="#">33.4.3/529</a>
4007_6003	SPI Control Register 1 (SPI0_C1)	8	R/W	04h	<a href="#">33.4.4/531</a>
4007_6004	SPI Match Register low (SPI0_ML)	8	R/W	00h	<a href="#">33.4.5/532</a>
4007_6005	SPI match register high (SPI0_MH)	8	R/W	00h	<a href="#">33.4.6/533</a>
4007_6006	SPI Data Register low (SPI0_DL)	8	R/W	00h	<a href="#">33.4.7/533</a>
4007_6007	SPI data register high (SPI0_DH)	8	R/W	00h	<a href="#">33.4.8/534</a>
4007_600A	SPI clear interrupt register (SPI0_CI)	8	R/W	00h	<a href="#">33.4.9/535</a>
4007_600B	SPI control register 3 (SPI0_C3)	8	R/W	00h	<a href="#">33.4.10/536</a>
4007_7000	SPI Status Register (SPI1_S)	8	R	20h	<a href="#">33.4.1/524</a>
4007_7001	SPI Baud Rate Register (SPI1_BR)	8	R/W	00h	<a href="#">33.4.2/528</a>
4007_7002	SPI Control Register 2 (SPI1_C2)	8	R/W	00h	<a href="#">33.4.3/529</a>
4007_7003	SPI Control Register 1 (SPI1_C1)	8	R/W	04h	<a href="#">33.4.4/531</a>
4007_7004	SPI Match Register low (SPI1_ML)	8	R/W	00h	<a href="#">33.4.5/532</a>
4007_7005	SPI match register high (SPI1_MH)	8	R/W	00h	<a href="#">33.4.6/533</a>
4007_7006	SPI Data Register low (SPI1_DL)	8	R/W	00h	<a href="#">33.4.7/533</a>
4007_7007	SPI data register high (SPI1_DH)	8	R/W	00h	<a href="#">33.4.8/534</a>
4007_700A	SPI clear interrupt register (SPI1_CI)	8	R/W	00h	<a href="#">33.4.9/535</a>
4007_700B	SPI control register 3 (SPI1_C3)	8	R/W	00h	<a href="#">33.4.10/536</a>

### 33.4.1 SPI Status Register (SPIx\_S)

This register contains read-only status bits.

**NOTE**

When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): Bits 3 through 0 are not implemented and always read 0.

When the FIFO is supported and enabled (FIFOMODE is 1): This register has four flags that provide mechanisms to support an 8-byte FIFO mode: RNFULLF, TNEARF, TXFULLF, and RFIFOEF. When the SPI is in 8-byte FIFO mode, the function of SPRF and SPTEF differs slightly from their function in the normal buffered modes, mainly regarding how these flags are cleared by the amount available in the transmit and receive FIFOs.

- The RNFULLF and TNEAREF help improve the efficiency of FIFO operation when transferring large amounts of data. These flags provide a "watermark" feature of the FIFOs to allow continuous transmissions of data when running at high speed.
- The RNFULLF can generate an interrupt if the RNFULLIEN bit in the C3 register is set, which allows the CPU to start emptying the receive FIFO without delaying the reception of subsequent bytes. The user can also determine if all data in the receive FIFO has been read by monitoring the RFIFOEF.
- The TNEAREF can generate an interrupt if the TNEARIEN bit in the C3 register is set, which allows the CPU to start filling the transmit FIFO before it is empty and thus to prevent breaks in SPI transmission.

**NOTE**

At an initial POR, the values of TNEAREF and RFIFOEF are 0. However, the status (S) register and both TX and RX FIFOs are reset due to a change of SPIMODE, FIFOMODE or SPE. If this type of reset occurs and FIFOMODE is 0, TNEAREF and RFIFOEF continue to reset to 0. If this type of reset occurs and FIFOMODE is 1, TNEAREF and RFIFOEF reset to 1.

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOEF
Write		w1c						
Reset	0	0	1	0	0	0	0	0

**SPIx\_S field descriptions**

Field	Description
7 SPRF	<p>SPI Read Buffer Full Flag (when FIFO is not supported or not enabled) or SPI read FIFO FULL flag (when FIFO is supported and enabled)</p> <p>When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data (DH:DL) register. When the receive DMA request is disabled (RXDMAE is 0), SPRF is cleared by reading SPRF while it is set and then reading the SPI data register. When the receive DMA request is enabled (RXDMAE</p>

*Table continues on the next page...*

## SPIx\_S field descriptions (continued)

Field	Description
	<p>is 1), SPRF is automatically cleared when the DMA transfer for the receive DMA request is completed (RX DMA Done is asserted).</p> <p>When FIFOMODE is 1: This bit indicates the status of the read FIFO when FIFOMODE is enabled. The SPRF is set when the read FIFO has received 64 bits (4 words or 8 bytes) of data from the shifter and there have been no CPU reads of the SPI data (DH:DL) register. When the receive DMA request is disabled (RXDMAE is 0), SPRF is cleared by reading the SPI data register, resulting in the FIFO no longer being full, assuming another SPI message is not received. When the receive DMA request is enabled (RXDMAE is 1), SPRF is automatically cleared when the first DMA transfer for the receive DMA request is completed (RX DMA Done is asserted).</p> <p>0 No data available in the receive data buffer (when FIFOMODE is not present or is 0) or Read FIFO is not full (when FIFOMODE is 1)</p> <p>1 Data available in the receive data buffer (when FIFOMODE is not present or is 0) or Read FIFO is full (when FIFOMODE is 1)</p>
6 SPMF	<p>SPI Match Flag</p> <p>SPMF is set after SPRF is 1 when the value in the receive data buffer matches the value in the MH:ML registers. To clear the flag, read SPMF when it is set and then write a 1 to it.</p> <p>0 Value in the receive data buffer does not match the value in the MH:ML registers</p> <p>1 Value in the receive data buffer matches the value in the MH:ML registers</p>
5 SPTEF	<p>SPI Transmit Buffer Empty Flag (when FIFO is not supported or not enabled) or SPI transmit FIFO empty flag (when FIFO is supported and enabled)</p> <p>When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): This bit is set when the transmit data buffer is empty. When the transmit DMA request is disabled (TXDMAE is 0), SPTEF is cleared by reading the S register with SPTEF set and then writing a data value to the transmit buffer at DH:DL. The S register must be read with SPTEF set to 1 before writing data to the DH:DL register; otherwise, the DH:DL write is ignored. When the transmit DMA request is enabled (TXDMAE is 1), SPTEF is automatically cleared when the DMA transfer for the transmit DMA request is completed (TX DMA Done is asserted). SPTEF is automatically set when all data from the transmit buffer transfers into the transmit shift register. For an idle SPI, data written to DH:DL is transferred to the shifter almost immediately so that SPTEF is set within two bus cycles, allowing a second set of data to be queued into the transmit buffer. After completion of the transfer of the data in the shift register, the queued data from the transmit buffer automatically moves to the shifter, and SPTEF is set to indicate that room exists for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter.</p> <p>When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): If a transfer does not stop, the last data that was transmitted is sent out again.</p> <p>When the FIFO is supported and enabled (FIFOMODE is 1): This bit provides the status of the FIFO rather than an 8-bit or a 16-bit buffer. This bit is set when the transmit FIFO is empty. When the transmit DMA request is disabled (TXDMAE is 0), SPTEF is cleared by writing a data value to the transmit FIFO at DH:DL. When the transmit DMA request is enabled (TXDMAE is 1), SPTEF is automatically cleared when the DMA transfer for the transmit DMA request is completed (TX DMA Done is asserted). SPTEF is automatically set when all data from the transmit FIFO transfers into the transmit shift register. For an idle SPI, data written to the DH:DL register is transferred to the shifter almost immediately, so that SPTEF is set within two bus cycles. A second write of data to the DH:DL register clears this SPTEF flag. After completion of the transfer of the data in the shift register, the queued data from the transmit FIFO automatically moves to the shifter, and SPTEF will be set only when all data written to the transmit FIFO has been transferred to the shifter. If no new data is waiting in the transmit FIFO, SPTEF simply remains set and no data moves from the buffer to the shifter.</p>

*Table continues on the next page...*

## SPIx\_S field descriptions (continued)

Field	Description
	<p>0 SPI transmit buffer not empty (when FIFOMODE is not present or is 0) or SPI FIFO not empty (when FIFOMODE is 1)</p> <p>1 SPI transmit buffer empty (when FIFOMODE is not present or is 0) or SPI FIFO empty (when FIFOMODE is 1)</p>
4 MODF	<p>Master Mode Fault Flag</p> <p>MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The <math>\overline{SS}</math> pin acts as a mode fault error input only when C1[MSTR] is 1, C2[MODFEN] is 1, and C1[SSOE] is 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1 and then writing to the SPI Control Register 1 (C1).</p> <p>0 No mode fault error</p> <p>1 Mode fault error detected</p>
3 RNFULLF	<p>Receive FIFO nearly full flag</p> <p>This flag is set when more than three 16-bit words or six 8-bit bytes of data remain in the receive FIFO, provided C3[RNFULLF_MARK] is 0, or when more than two 16-bit words or four 8-bit bytes of data remain in the receive FIFO, provided C3[RNFULLF_MARK] is 1. It has no function if FIFOMODE is not present or is 0.</p> <p>0 Receive FIFO has received less than 48 bits (when C3[RNFULLF_MARK] is 0) or less than 32 bits (when C3[RNFULLF_MARK] is 1)</p> <p>1 Receive FIFO has received data of an amount equal to or greater than 48 bits (when C3[RNFULLF_MARK] is 0) or 32 bits (when C3[RNFULLF_MARK] is 1)</p>
2 TNEAREF	<p>Transmit FIFO nearly empty flag</p> <p>This flag is set when only one 16-bit word or two 8-bit bytes of data remain in the transmit FIFO, provided C3[TNEAREF_MARK] is 0, or when only two 16-bit words or four 8-bit bytes of data remain in the transmit FIFO, provided C3[TNEAREF_MARK] is 1. If FIFOMODE is not enabled, ignore this bit.</p> <p><b>NOTE:</b> At an initial POR, the values of TNEAREF and RFIFOEF are 0. However, the status (S) register and both TX and RX FIFOs are reset due to a change of SPIMODE, FIFOMODE or SPE. If this type of reset occurs and FIFOMODE is 0, TNEAREF and RFIFOEF continue to reset to 0. If this type of reset occurs and FIFOMODE is 1, TNEAREF and RFIFOEF reset to 1.</p> <p>0 Transmit FIFO has more than 16 bits (when C3[TNEAREF_MARK] is 0) or more than 32 bits (when C3[TNEAREF_MARK] is 1) remaining to transmit</p> <p>1 Transmit FIFO has an amount of data equal to or less than 16 bits (when C3[TNEAREF_MARK] is 0) or 32 bits (when C3[TNEAREF_MARK] is 1) remaining to transmit</p>
1 TXFULLF	<p>Transmit FIFO full flag</p> <p>This bit indicates the status of the transmit FIFO when FIFOMODE is enabled. This flag is set when there are 8 bytes in the transmit FIFO. If FIFOMODE is not enabled, ignore this bit.</p> <p>When FIFOMODE and DMA are both enabled, the inverted TXFULLF is used to trigger a DMA transfer. So when the transmit FIFO is not full, the DMA request is active, and remains active until the FIFO is full.</p> <p>0 Transmit FIFO has less than 8 bytes</p> <p>1 Transmit FIFO has 8 bytes of data</p>
0 RFIFOEF	<p>SPI read FIFO empty flag</p> <p>This bit indicates the status of the read FIFO when FIFOMODE is enabled. If FIFOMODE is not enabled, ignore this bit.</p>

Table continues on the next page...

### SPIx\_S field descriptions (continued)

Field	Description
	When FIFOMODE and DMA are both enabled, the inverted RXIFOEFEF is used to trigger a DMA transfer. So when the receive FIFO is not empty, the DMA request is active, and remains active until the FIFO is empty.
	<b>NOTE:</b> At an initial POR, the values of TNEAREF and RFIFOEF are 0. However, the status (S) register and both TX and RX FIFOs are reset due to a change of SPIMODE, FIFOMODE or SPE. If this type of reset occurs and FIFOMODE is 0, TNEAREF and RFIFOEF continue to reset to 0. If this type of reset occurs and FIFOMODE is 1, TNEAREF and RFIFOEF reset to 1.
0	Read FIFO has data. Reads of the DH:DL registers in 16-bit mode or the DL register in 8-bit mode will empty the read FIFO.
1	Read FIFO is empty.

### 33.4.2 SPI Baud Rate Register (SPIx\_BR)

Use this register to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read	0	SPPR[2:0]			SPR[3:0]			
Write	0	0			0			
Reset	0	0	0	0	0	0	0	0

#### SPIx\_BR field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6-4 SPPR[2:0]	SPI Baud Rate Prescale Divisor  This 3-bit field selects one of eight divisors for the SPI baud rate prescaler. The input to this prescaler is the SPI module clock. The output of this prescaler drives the input of the SPI baud rate divider. Refer to the description of “SPI Baud Rate Generation” for details.  000 Baud rate prescaler divisor is 1. 001 Baud rate prescaler divisor is 2. 010 Baud rate prescaler divisor is 3. 011 Baud rate prescaler divisor is 4. 100 Baud rate prescaler divisor is 5. 101 Baud rate prescaler divisor is 6. 110 Baud rate prescaler divisor is 7. 111 Baud rate prescaler divisor is 8.
SPR[3:0]	SPI Baud Rate Divisor

Table continues on the next page...



### SPIx\_BR field descriptions (continued)

Field	Description
	This 4-bit field selects one of nine divisors for the SPI baud rate divider. The input to this divider comes from the SPI baud rate prescaler. Refer to the description of “SPI Baud Rate Generation” for details.
0000	Baud rate divisor is 2.
0001	Baud rate divisor is 4.
0010	Baud rate divisor is 8.
0011	Baud rate divisor is 16.
0100	Baud rate divisor is 32.
0101	Baud rate divisor is 64.
0110	Baud rate divisor is 128.
0111	Baud rate divisor is 256.
1000	Baud rate divisor is 512.
All others	Reserved

### 33.4.3 SPI Control Register 2 (SPIx\_C2)

This read/write register is used to control optional features of the SPI system.

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read	SPMIE	SPIMODE	TXDMAE	MODFEN	BIDIROE	RXDMAE	SPISWAI	SPC0
Write								
Reset	0	0	0	0	0	0	0	0

#### SPIx\_C2 field descriptions

Field	Description
7 SPMIE	<p>SPI Match Interrupt Enable</p> <p>This is the interrupt enable bit for the SPI receive data buffer hardware match (SPMF) function.</p> <p>0 Interrupts from SPMF inhibited (use polling) 1 When SPMF is 1, requests a hardware interrupt</p>
6 SPIMODE	<p>SPI 8-bit or 16-bit mode</p> <p>This bit allows the user to select either an 8-bit or 16-bit SPI data transmission length. In master mode, a change of this bit aborts a transmission in progress, forces the SPI system into an idle state, and resets all status bits in the S register. Refer to the description of “Data Transmission Length” for details.</p> <p>0 8-bit SPI shift register, match register, and buffers 1 16-bit SPI shift register, match register, and buffers</p>
5 TXDMAE	<p>Transmit DMA enable</p> <p>This bit enables a transmit DMA request. When this bit is set to 1, a transmit DMA request is asserted when both SPTEF and SPE are set, and the interrupt from SPTEF is disabled.</p>

*Table continues on the next page...*

## SPIx\_C2 field descriptions (continued)

Field	Description
	0 DMA request for transmit is disabled and interrupt from SPTEF is allowed 1 DMA request for transmit is enabled and interrupt from SPTEF is disabled
4 MODFEN	Master Mode-Fault Function Enable  When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used. For details, refer to the description of the SSOE bit in the C1 register.  0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	Bidirectional Mode Output Enable  When bidirectional mode is enabled because SPI pin control 0 (SPC0) is set to 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 is 0, BIDIROE has no meaning or effect.  0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
2 RXDMAE	Receive DMA enable  This is the enable bit for a receive DMA request. When this bit is set to 1, a receive DMA request is asserted when both SPRF and SPE are set, and the interrupt from SPRF is disabled.  0 DMA request for receive is disabled and interrupt from SPRF is allowed 1 DMA request for receive is enabled and interrupt from SPRF is disabled
1 SPISWAI	SPI Stop in Wait Mode  This bit is used for power conservation while the device is in Wait mode.  0 SPI clocks continue to operate in Wait mode. 1 SPI clocks stop when the MCU enters Wait mode.
0 SPC0	SPI Pin Control 0  Enables bidirectional pin configurations.  0 SPI uses separate pins for data input and data output (pin mode is normal). In master mode of operation: MISO is master in and MOSI is master out. In slave mode of operation: MISO is slave out and MOSI is slave in. 1 SPI configured for single-wire bidirectional operation (pin mode is bidirectional). In master mode of operation: MISO is not used by SPI; MOSI is master in when BIDIROE is 0 or master I/O when BIDIROE is 1. In slave mode of operation: MISO is slave in when BIDIROE is 0 or slave I/O when BIDIROE is 1; MOSI is not used by SPI.

### 33.4.4 SPI Control Register 1 (SPIx\_C1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

Address: Base address + 3h offset

Bit	7	6	5	4	3	2	1	0
Read	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Write								
Reset	0	0	0	0	0	1	0	0

#### SPIx\_C1 field descriptions

Field	Description
7 SPIE	<p>SPI Interrupt Enable: for SPRF and MODF (when FIFO is not supported or not enabled) or for read FIFO (when FIFO is supported and enabled)</p> <p>When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): Enables the interrupt for SPI receive buffer full (SPRF) and mode fault (MODF) events.</p> <p>When the FIFO is supported and enabled (FIFOMODE is 1): This bit enables the SPI to interrupt the CPU when the receive FIFO is full. An interrupt occurs when the SPRF bit is set or the MODF bit is set.</p> <p>0 Interrupts from SPRF and MODF are inhibited—use polling (when FIFOMODE is not present or is 0) or Read FIFO Full Interrupts are disabled (when FIFOMODE is 1)</p> <p>1 Request a hardware interrupt when SPRF or MODF is 1 (when FIFOMODE is not present or is 0) or Read FIFO Full Interrupts are enabled (when FIFOMODE is 1)</p>
6 SPE	<p>SPI System Enable</p> <p>Enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, the SPI is disabled and forced into an idle state, and all status bits in the S register are reset.</p> <p>0 SPI system inactive</p> <p>1 SPI system enabled</p>
5 SPTIE	<p>SPI Transmit Interrupt Enable</p> <p>When the FIFO is not supported or not enabled (FIFOMODE is not present or is 0): This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). An interrupt occurs when the SPI transmit buffer is empty (SPTEF is set).</p> <p>When the FIFO is supported and enabled (FIFOMODE is 1): This is the interrupt enable bit for SPI transmit FIFO empty (SPTEF). An interrupt occurs when the SPI transmit FIFO is empty (SPTEF is set).</p> <p>0 Interrupts from SPTEF inhibited (use polling)</p> <p>1 When SPTEF is 1, hardware interrupt requested</p>
4 MSTR	<p>Master/Slave Mode Select</p> <p>Selects master or slave mode operation.</p> <p>0 SPI module configured as a slave SPI device</p> <p>1 SPI module configured as a master SPI device</p>

Table continues on the next page...

## SPIx\_C1 field descriptions (continued)

Field	Description
3 CPOL	<p>Clock Polarity</p> <p>Selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values.</p> <p>This bit effectively places an inverter in series with the clock signal either from a master SPI device or to a slave SPI device. Refer to the description of “SPI Clock Formats” for details.</p> <p>0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)</p>
2 CPHA	<p>Clock Phase</p> <p>Selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to the description of “SPI Clock Formats” for details.</p> <p>0 First edge on SPSCCK occurs at the middle of the first cycle of a data transfer. 1 First edge on SPSCCK occurs at the start of the first cycle of a data transfer.</p>
1 SSOE	<p>Slave Select Output Enable</p> <p>This bit is used in combination with the Mode Fault Enable (MODFEN) field in the C2 register and the Master/Slave (MSTR) control bit to determine the function of the <math>\overline{SS}</math> pin.</p> <p>0 When C2[MODFEN] is 0: In master mode, <math>\overline{SS}</math> pin function is general-purpose I/O (not SPI). In slave mode, <math>\overline{SS}</math> pin function is slave select input.</p> <p>When C2[MODFEN] is 1: In master mode, <math>\overline{SS}</math> pin function is <math>\overline{SS}</math> input for mode fault. In slave mode, <math>\overline{SS}</math> pin function is slave select input.</p> <p>1 When C2[MODFEN] is 0: In master mode, <math>\overline{SS}</math> pin function is general-purpose I/O (not SPI). In slave mode, <math>\overline{SS}</math> pin function is slave select input.</p> <p>When C2[MODFEN] is 1: In master mode, <math>\overline{SS}</math> pin function is automatic <math>\overline{SS}</math> output. In slave mode: <math>\overline{SS}</math> pin function is slave select input.</p>
0 LSBFE	<p>LSB First (shifter direction)</p> <p>This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register always have the MSB in bit 7 (or bit 15 in 16-bit mode).</p> <p>0 SPI serial data transfers start with the most significant bit. 1 SPI serial data transfers start with the least significant bit.</p>

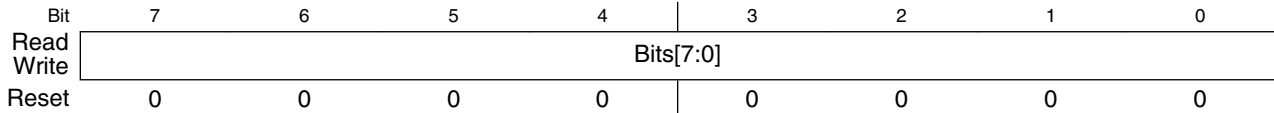
### 33.4.5 SPI Match Register low (SPIx\_ML)

This register, together with the MH register, contains the hardware compare value. When the value received in the SPI receive data buffer equals this hardware compare value, the SPI Match Flag in the S register (S[SPMF]) sets.

In 8-bit mode, only the ML register is available. Reads of the MH register return all zeros. Writes to the MH register are ignored.

In 16-bit mode, reading either byte (the MH or ML register) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (the MH or ML register) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent value into the SPI match registers.

Address: Base address + 4h offset



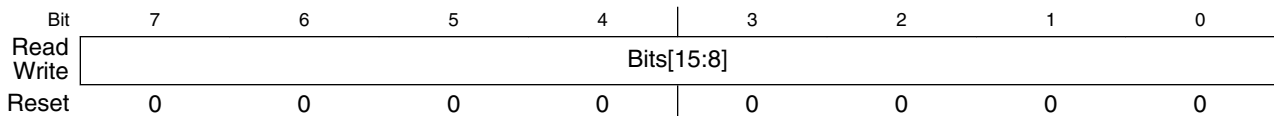
#### SPIx\_ML field descriptions

Field	Description
Bits[7:0]	Hardware compare value (low byte)

### 33.4.6 SPI match register high (SPIx\_MH)

Refer to the description of the ML register.

Address: Base address + 5h offset



#### SPIx\_MH field descriptions

Field	Description
Bits[15:8]	Hardware compare value (high byte)

### 33.4.7 SPI Data Register low (SPIx\_DL)

This register, together with the DH register, is both the input and output register for SPI data. A write to the registers writes to the transmit data buffer, allowing data to be queued and transmitted.

When the SPI is configured as a master, data queued in the transmit data buffer is transmitted immediately after the previous transmission has completed.

The SPTEF bit in the S register indicates when the transmit data buffer is ready to accept new data. When the transmit DMA request is disabled (TXDMAE is 0): The S register must be read when S[SPTEF] is set before writing to the SPI data registers; otherwise, the

## Memory map/register definition

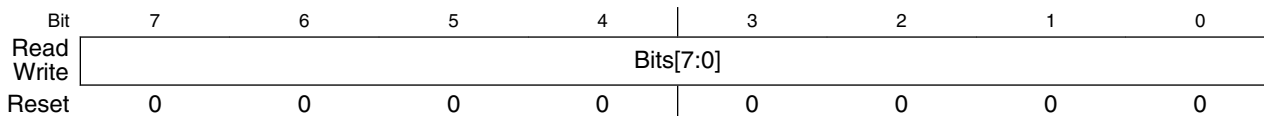
write is ignored. When the transmit DMA request is enabled (TXDMAE is 1) when S[SPTEF] is set, the SPI data registers can be written automatically by DMA without reading the S register first.

Data may be read from the SPI data registers any time after S[SPRF] is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition, and the data from the new transfer is lost. The new data is lost because the receive buffer still held the previous character and was not ready to accept the new data. There is no indication for a receive overrun condition, so the application system designer must ensure that previous data has been read from the receive buffer before a new transfer is initiated.

In 8-bit mode, only the DL register is available. Reads of the DH register return all zeros. Writes to the DH register are ignored.

In 16-bit mode, reading either byte (the DH or DL register) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (the DH or DL register) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

Address: Base address + 6h offset



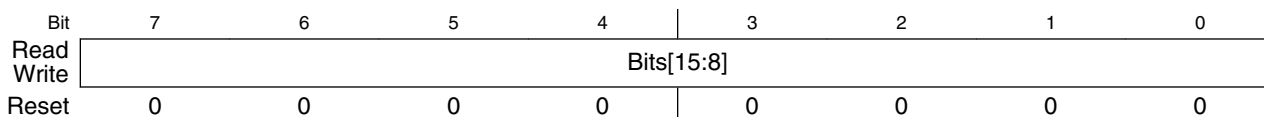
### SPIx\_DL field descriptions

Field	Description
Bits[7:0]	Data (low byte)

## 33.4.8 SPI data register high (SPIx\_DH)

Refer to the description of the DL register.

Address: Base address + 7h offset



### SPIx\_DH field descriptions

Field	Description
Bits[15:8]	Data (high byte)

### 33.4.9 SPI clear interrupt register (SPIx\_CI)

This register applies only for an instance of the SPI module that supports the FIFO feature.

The register has four bits dedicated to clearing the interrupts. Writing 1 to these bits clears the corresponding interrupts if the INTCLR bit in the C3 register is 1. Reading these bits always returns 0.

This register also has two read-only bits to indicate the transmit FIFO and receive FIFO overrun conditions. When the receive FIFO is full and data is received, RXFOF is set. Similarly, when the transmit FIFO is full and a write to the data register occurs, TXFOF is set. These flags are cleared when the CI register is read while the flags are set.

The register has two more read-only bits to indicate the error flags. These flags are set when, due to some spurious reason, entries in the FIFO total more than 64 bits of data. At this point, all the flags in the status register are reset, and entries in the FIFO are flushed with the corresponding error flags set. These flags are cleared when the CI register is read while the flags are set.

Address: Base address + Ah offset

Bit	7	6	5	4	3	2	1	0
Read	TXFERR	RXFERR	TXFOF	RXFOF	0	0	0	0
Write					TNEAREFCI	RNFULLFCI	SPTEFCI	SPRFCI
Reset	0	0	0	0	0	0	0	0

#### SPIx\_CI field descriptions

Field	Description
7 TXFERR	Transmit FIFO error flag This flag indicates that a transmit FIFO error occurred because entries in the FIFO total more than 64 bits of data. 0 No transmit FIFO error occurred 1 A transmit FIFO error occurred
6 RXFERR	Receive FIFO error flag This flag indicates that a receive FIFO error occurred because entries in the FIFO total more than 64 bits of data. 0 No receive FIFO error occurred 1 A receive FIFO error occurred
5 TXFOF	Transmit FIFO overflow flag This flag indicates that a transmit FIFO overflow condition has occurred.

*Table continues on the next page...*

**SPIx\_CI field descriptions (continued)**

Field	Description
	0 Transmit FIFO overflow condition has not occurred 1 Transmit FIFO overflow condition occurred
4 RXFOF	Receive FIFO overflow flag This flag indicates that a receive FIFO overflow condition has occurred. 0 Receive FIFO overflow condition has not occurred 1 Receive FIFO overflow condition occurred
3 TNEAREFCI	Transmit FIFO nearly empty flag clear interrupt Writing 1 to this bit clears the TNEAREF interrupt provided that C3[3] is set.
2 RNFULLFCI	Receive FIFO nearly full flag clear interrupt Writing 1 to this bit clears the RNFULLF interrupt provided that C3[3] is set.
1 SPTEFCI	Transmit FIFO empty flag clear interrupt Writing 1 to this bit clears the SPTEF interrupt provided that C3[3] is set.
0 SPRFCI	Receive FIFO full flag clear interrupt Writing 1 to this bit clears the SPRF interrupt provided that C3[3] is set.

**33.4.10 SPI control register 3 (SPIx\_C3)**

This register introduces a 64-bit FIFO function on both transmit and receive buffers. It applies only for an instance of the SPI module that supports the FIFO feature.

FIFO mode is enabled by setting the FIFOMODE bit to 1. A write to this register occurs only when it sets the FIFOMODE bit to 1.

Using this FIFO feature allows the SPI to provide high speed transfers of large amounts of data without consuming large amounts of the CPU bandwidth.

Enabling this FIFO function affects the behavior of some of the read/write buffer flags in the S register as follows:

- When the receive FIFO has data in it, S[RFIFOEF] is 0. As a result:
  - If C2[RXDMAE] is 1, RFIFOEF\_b generates a receive DMA request. The DMA request remains active until RFIFOEF is set to 1, indicating the receive buffer is empty.
  - If C2[RXDMAE] is 0 and C1[SPIE] is 1, SPRF interrupts the CPU.
- When the transmit FIFO is not full, S[TXFULLF] is 0. As a result:
  - If C2[TXDMAE] is 1, TXFULLF\_b generates a transmit DMA request. The DMA request remains active until TXFULLF is set to 1, indicating the transmit FIFO is full.
  - If C2[TXDMAE] is 0 and C1[SPTIE] is 1, SPTEF interrupts the CPU.



Two interrupt enable bits, TNEARIEN and RNFULLIEN, provide CPU interrupts based on the "watermark" feature of the TNEARF and RNFULLF flags of the S register.

Address: Base address + Bh offset

Bit	7	6	5	4	3	2	1	0
Read	0		TNEAREF_	RNFULLF_	INTCLR	TNEARIEN	RNFULLIEN	FIFOMODE
Write			MARK	MARK				
Reset	0	0	0	0	0	0	0	0

### SPIx\_C3 field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 TNEAREF_	Transmit FIFO nearly empty watermark This bit selects the mark after which the TNEAREF flag is asserted. 0 TNEAREF is set when the transmit FIFO has 16 bits or less 1 TNEAREF is set when the transmit FIFO has 32 bits or less
4 RNFULLF_	Receive FIFO nearly full watermark This bit selects the mark after which the RNFULLF flag is asserted. 0 RNFULLF is set when the receive FIFO has 48 bits or more 1 RNFULLF is set when the receive FIFO has 32 bits or more
3 INTCLR	Interrupt clearing mechanism select This bit selects the mechanism by which the SPRF, SPTEF, TNEAREF, and RNFULLF interrupts are cleared. 0 These interrupts are cleared when the corresponding flags are cleared depending on the state of the FIFOs 1 These interrupts are cleared by writing the corresponding bits in the CI register
2 TNEARIEN	Transmit FIFO nearly empty interrupt enable Writing 1 to this bit enables the SPI to interrupt the CPU when the TNEAREF flag is set. This bit is ignored and has no function if the FIFOMODE bit is 0. 0 No interrupt upon TNEAREF being set 1 Enable interrupts upon TNEAREF being set
1 RNFULLIEN	Receive FIFO nearly full interrupt enable Writing 1 to this bit enables the SPI to interrupt the CPU when the RNFULLF flag is set. This bit is ignored and has no function if the FIFOMODE bit is 0. 0 No interrupt upon RNFULLF being set 1 Enable interrupts upon RNFULLF being set
0 FIFOMODE	FIFO mode enable This bit enables the SPI to use a 64-bit FIFO (8 bytes or four 16-bit words) for both transmit and receive buffers. 0 FIFO mode disabled 1 FIFO mode enabled

## 33.5 Functional description

This section provides the functional description of the module.

### 33.5.1 General

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI Control Register 1. While C1[SPE] is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select (SS)
- Serial clock (SPSCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

An SPI transfer is initiated in the master SPI device by reading the SPI status register (SPIx\_S) when S[SPTEF] = 1 and then writing data to the transmit data buffer (write to SPIx\_DH:SPIx\_DL). When a transfer is complete, received data is moved into the receive data buffer. The SPIx\_DH:SPIx\_DL registers act as the SPI receive data buffer for reads and as the SPI transmit data buffer for writes.

The Clock Phase Control (CPHA) and Clock Polarity Control (CPOL) bits in the SPI Control Register 1 (SPIx\_C1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. C1[CPHA] is used to accommodate two fundamentally different protocols by sampling data on odd numbered SPSCK edges or on even numbered SPSCK edges.

The SPI can be configured to operate as a master or as a slave. When the MSTR bit in SPI Control Register 1 is set, master mode is selected; when C1[MSTR] is clear, slave mode is selected.

### 33.5.2 Master mode

The SPI operates in master mode when C1[MSTR] is set. Only a master SPI module can initiate transmissions. A transmission begins by reading the SPIx\_S register while S[SPTEF] = 1 and writing to the master SPI data registers. If the shift register is empty, the byte immediately transfers to the shift register. The data begins shifting out on the MOSI pin under the control of the serial clock.

- **SPSCK**
  - The SPR3, SPR2, SPR1, and SPR0 baud rate selection bits in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI Baud Rate register control the baud rate generator and determine the speed of the transmission. The SPSCK pin is the SPI clock output. Through the SPSCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.
- **MOSI, MISO pin**
  - In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.
- **$\overline{SS}$  pin**
  - If C2[MODFEN] and C1[SSOE] are set, the SS pin is configured as slave select output. The SS output becomes low during each transmission and is high when the SPI is in idle state. If C2[MODFEN] is set and C1[SSOE] is cleared, the  $\overline{SS}$  pin is configured as input for detecting mode fault error. If the SS input becomes low this indicates a mode fault error where another master tries to drive the MOSI and SPSCK lines. In this case, the SPI immediately switches to slave mode by clearing C1[MSTR] and also disables the slave output buffer MISO (or SISO in bidirectional mode). As a result, all outputs are disabled, and SPSCK, MOSI and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state. This mode fault error also sets the Mode Fault (MODF) flag in the SPI Status Register (SPIx\_S). If the SPI Interrupt Enable bit (SPIE) is set when S[MODF] gets set, then an SPI interrupt sequence is also requested. When a write to the SPI Data Register in the master occurs, there is a half SPSCK-cycle delay. After the delay, SPSCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI Control Register 1 (see [SPI clock formats](#)).

### Note

A change of C1[CPOL], C1[CPHA], C1[SSOE], C1[LSBFE], C2[MODFEN], C2[SPC0], C2[BIDIROE] with C2[SPC0] set, SPIMODE, FIFOMODE, SPPR2-SPPR0 and SPR3-SPR0 in master mode abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master has to ensure that the remote slave is set back to idle state.

### 33.5.3 Slave mode

The SPI operates in slave mode when the MSTR bit in SPI Control Register 1 is clear.

- SPSCCK

In slave mode, SPSCCK is the SPI clock input from the master.

- MISO, MOSI pin

In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit and BIDIROE bit in SPI Control Register 2.

- SS pin

The SS pin is the slave select input. Before a data transmission occurs, the SS pin of the slave SPI must be low. SS must remain low until the transmission is complete. If SS goes high, the SPI is forced into an idle state.

The SS input also controls the serial data output pin. If SS is high (not selected), the serial data output pin is high impedance. If SS is low, the first bit in the SPI Data Register is driven out of the serial data output pin. Also, if the slave is not selected (SS is high), then the SPSCCK input is ignored and no internal shifting of the SPI shift register occurs.

Although the SPI is capable of duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin.

#### Note

When peripherals with duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave's serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI Control Register 1 is clear, odd numbered edges on the SPSCCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If C1[CPHA] is set, even numbered edges on the SPSCCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on C1[LSBFE].

When C1[CPHA] is set, the first edge is used to get the first data bit onto the serial data output pin. When C1[CPHA] is clear and the SS input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the eighth (SPIMODE = 0) or sixteenth (SPIMODE = 1) shift, the transfer is considered complete and the received data is transferred into the SPI Data register. To indicate transfer is complete, the SPRF flag in the SPI Status Register is set.

### Note

A change of the bits FIFOMODE, SPIMODE, C2[BIDIROE] with C2[SPC0] set, C1[CPOL], C1[CPHA], C1[SSOE], C1[LSBFE], C2[MODFEN], and C2[SPC0] in slave mode will corrupt a transmission in progress and must be avoided.

## 33.5.4 SPI FIFO Mode

When the FIFO feature is supported: The SPI works in FIFO mode when the C3[FIFOMODE] bit is set. When the module is in FIFO mode, the SPI RX buffer and SPI TX buffer are replaced by an 8-byte-deep FIFO, as the following figures show.

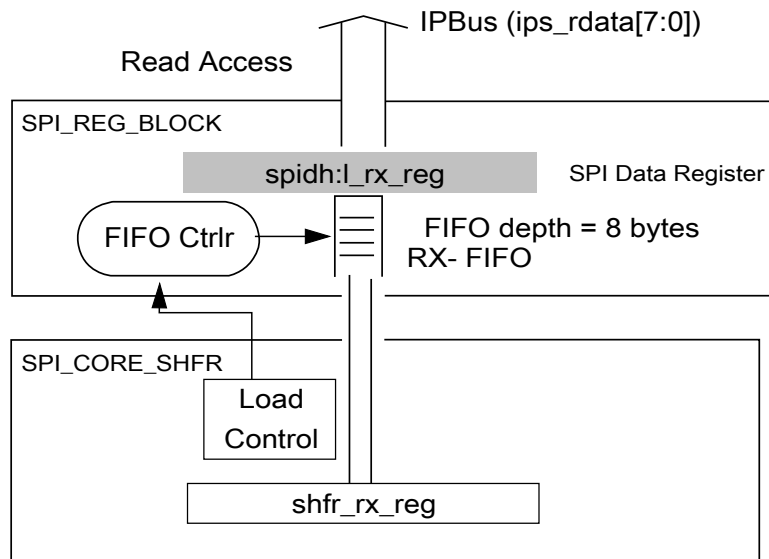


Figure 33-4. SPIH:L read side structural overview in FIFO mode

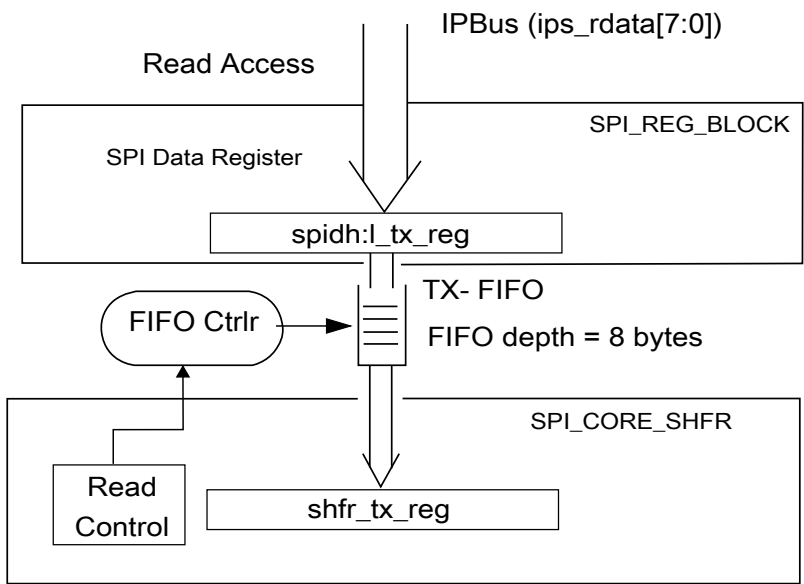


Figure 33-5. SPIH:L write side structural overview in FIFO mode

### 33.5.5 SPI Transmission by DMA

SPI supports both Transmit and Receive by DMA. The basic flow of SPI transmission by DMA is as below.

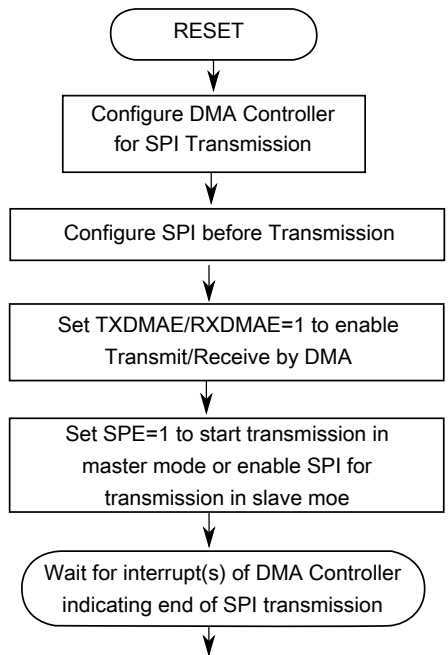


Figure 33-6. Basic Flow of SPI Transmission by DMA

### 33.5.5.1 Transmit by DMA

Transmit by DMA is supported only when TXDMAE is set. A transmit DMA request is asserted when both SPE and SPTEF are set. Then the on-chip DMA controller detects this request and transfers data from memory into the SPI data register. After that, TX DMA DONE is asserted to clear SPTEF automatically. This process repeats until all data for transmission (the number is decided by the configuration register[s] of the DMA controller) is sent.

When the FIFO feature is supported: In FIFO mode (FIFOMODE=1) and when a data length of 8 bits is selected (SPIMODE=0), the DMA transfer for one transmit DMA request can write more than 1 byte (up to 8 bytes) to the DL register because the TX FIFO can store 8 bytes of transmit data. In FIFO mode (FIFOMODE=1) and when a data length of 16 bits is selected (SPIMODE=1), the DMA transfer for one transmit DMA request can write more than 1 word (up to 4 words) to the DH:DL registers because the TX FIFO can store 4 words of transmit data. A larger number of bytes or words transferred from memory to the SPI data register for each transmit DMA request results in a lower total number of transmit DMA requests.

When FIFOMODE is 0: Cycle Steal (DMA\_DCRn[CS] = 1) should be enabled when using the DMA controller to transfer data from memory to the SPI data register. The DMA performs a single data transfer per DMA request in cycle steal mode. Therefore, a single byte/word is written to the SPI data register from memory and transmitted by the SPI module for each DMA request, as long as the BCR value is greater than zero (DMA\_DSR\_BCRn[BCR] > 0). Once the BCR has reached zero, software must reconfigure the DMA controller if more data is to be transmitted. If a configuration error occurs (DMA\_DSR\_BCRn[CE] = 1) when the BCR is equal to 0, software must:

- disable peripheral requests when the BCR is equal to 0,
- perform 16-bit transfers (SPIMODE = 1), or
- decrease the SPI baud rate.

Software can disable peripheral requests by setting DMA\_DCRn[D\_REQ] = 1 when initializing the DMA controller, or by clearing DMA\_DCRn[ERQ] once the BCR is equal to zero. Also, to continue transmitting data software must re-enable peripheral requests (DMA\_DCRn[ERQ] = 1) after reconfiguring the DMA controller.

### 33.5.5.2 Receive by DMA

Receive by DMA is supported only when RXDMAE is set. A receive DMA request is asserted when both SPE and SPRF are set. Then the on-chip DMA controller detects this request and transfers data from the SPI data register into memory. After that, RX DMA DONE is asserted to clear SPRF automatically. This process repeats until all data to be

received (the number is decided by configuration register[s] of the DMA controller) is received or no receive DMA request is generated again because the SPI transmission is finished.

When the FIFO feature is supported: In FIFO mode (FIFOMODE=1) and when a data length of 8 bits is selected (SPIMODE=0), the DMA transfer for one receive DMA request can read more than 1 byte (up to 8 bytes) from the SPI data register because the RX FIFO can hold 8 bytes. In FIFO mode (FIFOMODE=1) and when a data length of 16 bits is selected (SPIMODE=1), the DMA transfer for one receive DMA request can read more than 1 word (up to 4 words) from the DH:DL registers because the RX FIFO can hold 4 words. A larger number of bytes or words transferred from the SPI data register to memory for one receive DMA request results in a lower total number of receive DMA requests.

### **33.5.6 Data Transmission Length**

The SPI can support data lengths of 8 or 16 bits. The length can be configured with the SPIMODE bit in the SPIx\_C2 register.

In 8-bit mode (SPIMODE = 0), the SPI Data Register is comprised of one byte: SPIx\_DL. The SPI Match Register is also comprised of only one byte: SPIx\_ML. Reads of SPIx\_DH and SPIx\_MH will return zero. Writes to SPIx\_DH and SPIx\_MH will be ignored.

In 16-bit mode (SPIMODE = 1), the SPI Data Register is comprised of two bytes: SPIx\_DH and SPIx\_DL. Reading either byte (SPIx\_DH or SPIx\_DL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIx\_DH or SPIx\_DL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

In 16-bit mode, the SPI Match Register is also comprised of two bytes: SPIx\_MH and SPIx\_ML. There is no buffer mechanism for the reading of SPIxMH and SPIxML since they can only be changed by writing at CPU side. Writing to either byte (SPIx\_MH or SPIx\_ML) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the SPI Match Register.

Any switching between 8- and 16-bit data transmission length (controlled by SPIMODE bit) in master mode will abort a transmission in progress, force the SPI system into idle state, and reset all status bits in the SPIx\_S register. To initiate a transfer after writing to SPIMODE, the SPIx\_S register must be read with SPTEF = 1, and data must be written to SPIx\_DH:SPIx\_DL in 16-bit mode (SPIMODE = 1) or SPIx\_DL in 8-bit mode (SPIMODE = 0).



In slave mode, user software should write to SPIMODE only once to prevent corrupting a transmission in progress.

### Note

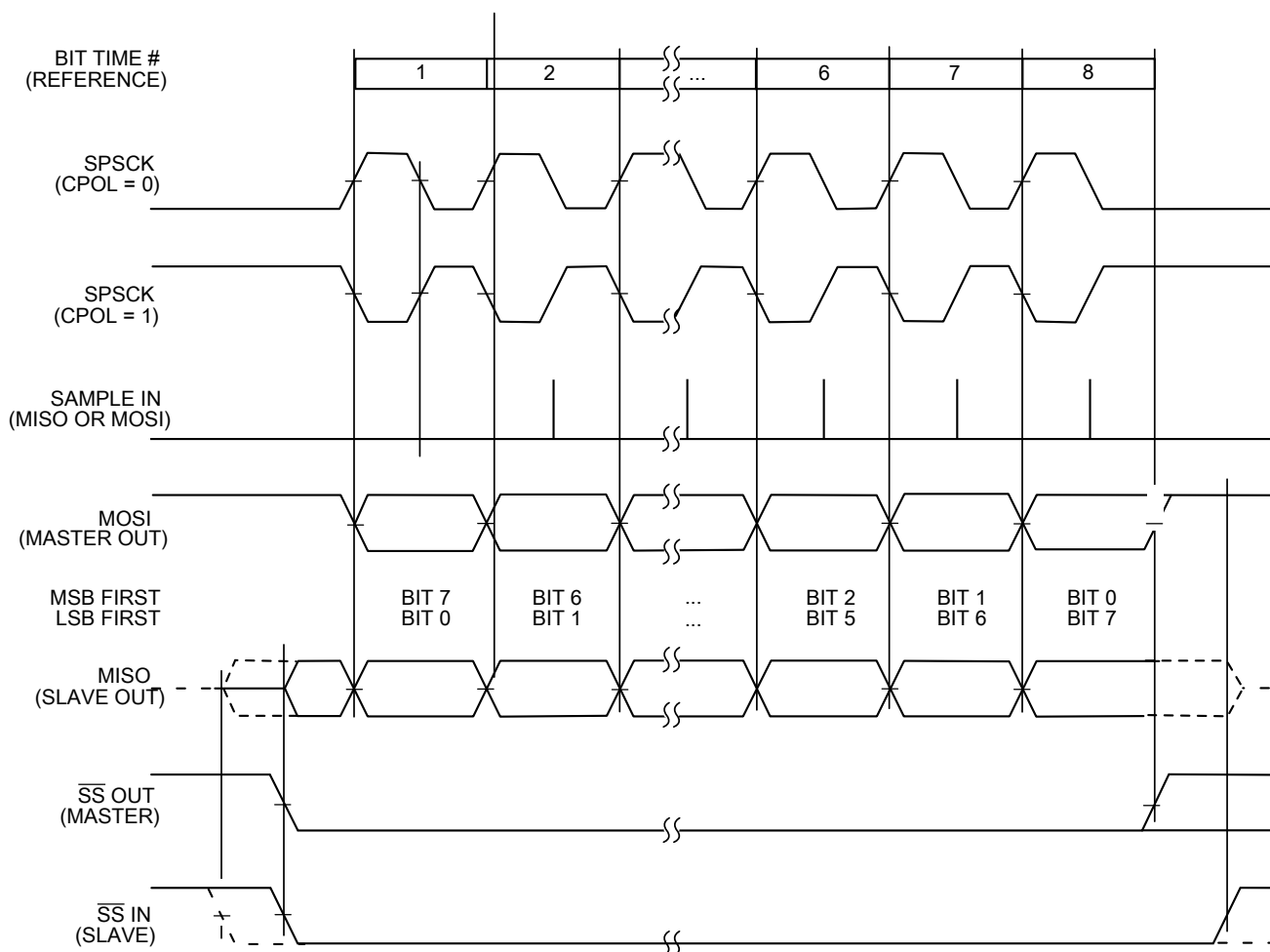
Data can be lost if the data length is not the same for both master and slave devices.

## 33.5.7 SPI clock formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a Clock Polarity (CPOL) bit and a Clock Phase (CPHA) control bit in the Control Register 1 to select one of four clock formats for data transfers. C1[CPOL] selectively inserts an inverter in series with the clock. C1[CPHA] chooses between two different clock phase relationships between the clock and data.

The following figure shows the clock formats when SPIMODE = 0 (8-bit mode) and CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCCK edge and bit 8 ending one-half SPSCCK cycle after the eighth SPSCCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in C1[CPOL]. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The  $\overline{SS}$  OUT waveform applies to the slave select output from a master (provided C2[MODFEN] and C1[SSOE] = 1). The master  $\overline{SS}$  output goes to active low one-half SPSCCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The  $\overline{SS}$  IN waveform applies to the slave select input of a slave.

## Functional description



**Figure 33-7. SPI clock formats (CPHA = 1)**

When  $C1[CPHA] = 1$ , the slave begins to drive its MISO output when  $\overline{SS}$  goes to active low, but the data is not defined until the first SPSCCK edge. The first SPSCCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively.

When  $C1[CPHA] = 1$ , the slave's  $\overline{SS}$  input is not required to go to its inactive high level between transfers. In this clock format, a back-to-back transmission can occur, as follows:

1. A transmission is in progress.
2. A new data byte is written to the transmit buffer before the in-progress transmission is complete.
3. When the in-progress transmission is complete, the new, ready data byte is transmitted immediately.

Between these two successive transmissions, no pause is inserted; the  $\overline{SS}$  pin remains low.

The following figure shows the clock formats when  $SPIMODE = 0$  and  $C1[CPHA] = 0$ . At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ( $\overline{SS}$  IN goes low), and bit 8 ends at the last SPSCCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in  $LSBFE$ . Both variations of SPSCCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in  $CPOL$ . The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The  $\overline{SS}$  OUT waveform applies to the slave select output from a master (provided  $C2[MODFEN]$  and  $C1[SSOE] = 1$ ). The master  $\overline{SS}$  output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCCK cycle after the end of the eighth bit time of the transfer. The  $\overline{SS}$  IN waveform applies to the slave select input of a slave.

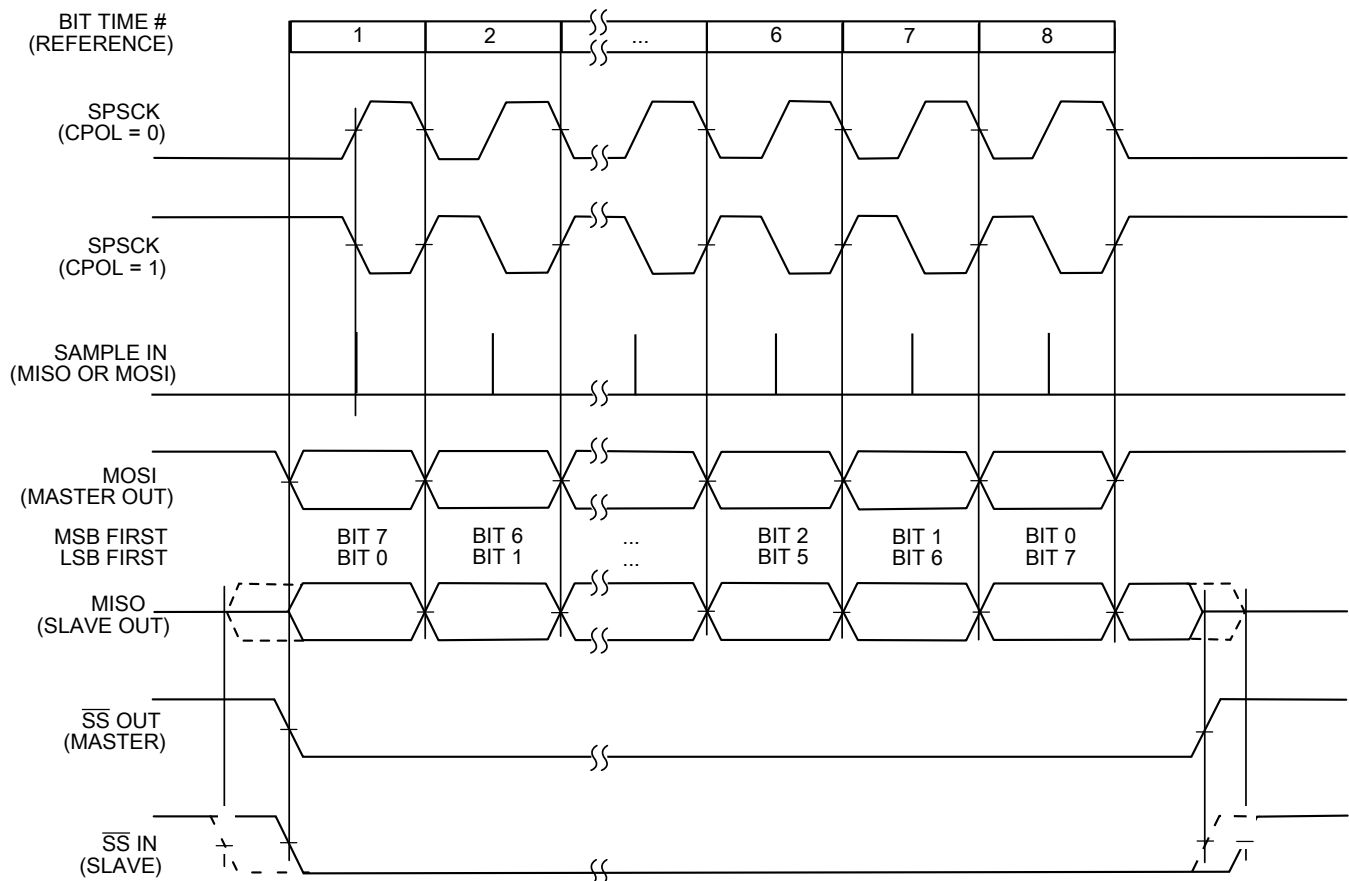


Figure 33-8. SPI clock formats (CPHA = 0)

When C1[CPHA] = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when SS goes to active low. The first SPSCCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When C1[CPHA] = 0, the slave's SS input must go to its inactive high level between transfers.

### 33.5.8 SPI baud rate generation

As shown in the following figure, the clock source for the SPI baud rate generator is the SPI module clock. The prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The rate-select bits (SPR3:SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, 256, or 512 to get the internal SPI master mode bit-rate clock.

The baud rate generator is activated only when the SPI is in the master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease I<sub>DD</sub> current.

The baud rate divisor equation is as follows (except those reserved combinations in the SPI Baud Rate Divisor table).

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \times 2^{(\text{SPR} + 1)}$$

The baud rate can be calculated with the following equation:

$$\text{BaudRate} = \text{SPI Module Clock} / \text{BaudRateDivisor}$$

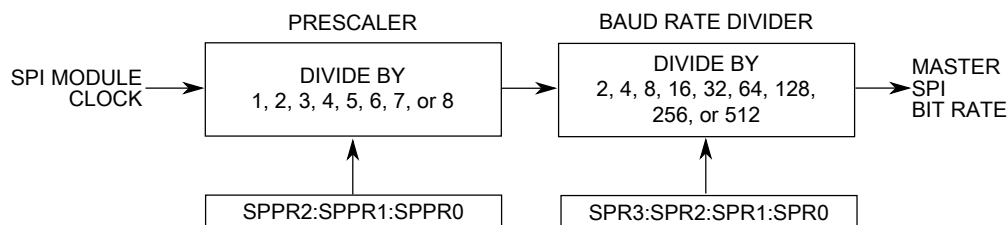


Figure 33-9. SPI baud rate generation

### 33.5.9 Special features

The following section describes the special features of SPI module.

### 33.5.9.1 $\overline{SS}$ Output

The  $\overline{SS}$  output feature automatically drives the  $\overline{SS}$  pin low during transmission to select external devices and drives the  $\overline{SS}$  pin high during idle to deselect external devices. When the  $\overline{SS}$  output is selected, the  $\overline{SS}$  output pin is connected to the  $\overline{SS}$  input pin of the external device.

The  $\overline{SS}$  output is available only in master mode during normal SPI operation by asserting C1[SSOE] and C2[MODFEN] as shown in the description of C1[SSOE].

The mode fault feature is disabled while  $\overline{SS}$  output is enabled.

#### Note

Be careful when using the  $\overline{SS}$  output feature in a multimaster system because the mode fault feature is not available for detecting system errors between masters.

### 33.5.9.2 Bidirectional mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI Control Register 2 (see the following table). In this mode, the SPI uses only one serial data pin for the interface with one or more external devices. C1[MSTR] decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The MISO pin in master mode and MOSI pin in slave mode are not used by the SPI.

**Table 33-2. Normal Mode and Bidirectional Mode**

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
<b>Normal Mode</b> SPC0 = 0		
<b>Bidirectional Mode</b> SPC0 = 1		

The direction of each serial I/O pin depends on C2[BIDIROE]. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

The SPSCCK is an output for the master mode and an input for the slave mode.

$\overline{SS}$  is the input or output for the master mode, and it is always the input for the slave mode.

The bidirectional mode does not affect SPSCCK and  $\overline{SS}$  functions.

### **Note**

In bidirectional master mode, with the mode fault feature enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode. In this case, MISO becomes occupied by the SPI and MOSI is not used. Consider this scenario if the MISO pin is used for another purpose.

## **33.5.10 Error conditions**

The SPI module has one error condition: the mode fault error.

### **33.5.10.1 Mode fault error**

If the  $\overline{SS}$  input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SPSCCK lines simultaneously. This condition is not permitted in normal operation, and it sets the MODF bit in the SPI status register automatically provided that C2[MODFEN] is set.

In the special case where the SPI is in master mode and C2[MODFEN] is cleared, the  $\overline{SS}$  pin is not used by the SPI. In this special case, the mode fault error function is inhibited and MODF remains cleared. If the SPI system is configured as a slave, the  $\overline{SS}$  pin is a dedicated input pin. A mode fault error does not occur in slave mode.

If a mode fault error occurs, the SPI is switched to slave mode, with the exception that the slave output buffer is disabled. So the SPSCCK, MISO and MOSI pins are forced to be high impedance inputs to avoid any possibility of conflict with another output driver. A transmission in progress is aborted and the SPI is forced into idle state.

If the mode fault error occurs in the bidirectional mode for an SPI system configured in master mode, the output enable of MOMI (MOSI in bidirectional mode) is cleared if it was set. No mode fault error occurs in the bidirectional mode for the SPI system configured in slave mode.

The mode fault flag is cleared automatically by a read of the SPI Status Register (with MODF set) followed by a write to SPI Control Register 1. If the mode fault flag is cleared, the SPI becomes a normal master or slave again.

### 33.5.11 Low-power mode options

This section describes the low-power mode options.

#### 33.5.11.1 SPI in Run mode

In Run mode, with the SPI system enable (SPE) bit in the SPI Control Register 1 clear, the SPI system is in a low-power, disabled state. SPI registers can still be accessed, but clocks to the core of this module are disabled.

#### 33.5.11.2 SPI in Wait mode

SPI operation in Wait mode depends upon the state of the SPISWAI bit in SPI Control Register 2.

- If C2[SPISWAI] is clear, the SPI operates normally when the CPU is in Wait mode.
- If C2[SPISWAI] is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
  - If C2[SPISWAI] is set and the SPI is configured for master, any transmission and reception in progress stops at Wait mode entry. The transmission and reception resumes when the SPI exits Wait mode.
  - If C2[SPISWAI] is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SPSCCK continues to be driven from the master. This keeps the slave synchronized to the master and the SPSCCK.

If the master transmits data while the slave is in wait mode, the slave continues to send data consistent with the operation mode at the start of wait mode (that is, if the slave is currently sending its SPIx\_DH:SPIx\_DL to the master, it continues to send the same byte. Otherwise, if the slave is currently sending the last data received byte from the master, it continues to send each previously received data from the master byte).

## Note

Care must be taken when expecting data from a master while the slave is in a Wait mode or a Stop mode where the peripheral bus clock is stopped but internal logic states are retained. Even though the shift register continues to operate, the rest of the SPI is shut down (that is, an SPRF interrupt is not generated until an exit from Stop or Wait mode). Also, the data from the shift register is not copied into the SPIx\_DH:SPIx\_DL registers until after the slave SPI has exited Wait or Stop mode. An SPRF flag and SPIx\_DH:SPIx\_DL copy is only generated if Wait mode is entered or exited during a transmission. If the slave enters Wait mode in idle mode and exits Wait mode in idle mode, neither an SPRF nor a SPIx\_DH:SPIx\_DL copy occurs.

### 33.5.11.3 SPI in Stop mode

Operation in a Stop mode where the peripheral bus clock is stopped but internal logic states are retained depends on the SPI system. The Stop mode does not depend on C2[SPISWAI]. Upon entry to this type of stop mode, the SPI module clock is disabled (held high or low).

- If the SPI is in master mode and exchanging data when the CPU enters the Stop mode, the transmission is frozen until the CPU exits stop mode. After the exit from stop mode, data to and from the external SPI is exchanged correctly.
- In slave mode, the SPI remains synchronized with the master.

The SPI is completely disabled in a stop mode where the peripheral bus clock is stopped and internal logic states are not retained. After an exit from this type of stop mode, all registers are reset to their default values, and the SPI module must be reinitialized.

### 33.5.12 Reset

The reset values of registers and signals are described in the Memory Map and Register Descriptions content, which details the registers and their bitfields.

- If a data transmission occurs in slave mode after a reset without a write to SPIx\_DH:SPIx\_DL, the transmission consists of "garbage" or the data last received from the master before the reset.
- Reading from SPIx\_DH:SPIx\_DL after reset always returns zeros.



### 33.5.13 Interrupts

The SPI originates interrupt requests only when the SPI is enabled (the SPE bit in the SPIx\_C1 register is set). The following is a description of how the SPI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt priority are chip dependent.

Four flag bits, three interrupt mask bits, and one interrupt vector are associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). The SPI match interrupt enable mask bit (SPIMIE) enables interrupts from the SPI match flag (SPMF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check the flag bits to determine which event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

#### 33.5.13.1 MODF

MODF occurs when the master detects an error on the  $\overline{SS}$  pin. The master SPI must be configured for the MODF feature (see the description of the C1[SSOE] bit). Once MODF is set, the current transfer is aborted and the master (MSTR) bit in the SPIx\_C1 register resets to 0.

The MODF interrupt is reflected in the status register's MODF flag. Clearing the flag also clears the interrupt. This interrupt stays active while the MODF flag is set. MODF has an automatic clearing process that is described in the SPI Status Register.

#### 33.5.13.2 SPRF

SPRF occurs when new data has been received and copied to the SPI receive data buffer. In 8-bit mode, SPRF is set only after all 8 bits have been shifted out of the shift register and into SPIx\_DL. In 16-bit mode, SPRF is set only after all 16 bits have been shifted out of the shift register and into SPIx\_DH:SPIx\_DL.

After SPRF is set, it does not clear until it is serviced. SPRF has an automatic clearing process that is described in the SPI Status Register details. If the SPRF is not serviced before the end of the next transfer (that is, SPRF remains active throughout another transfer), the subsequent transfers are ignored and no new data is copied into the Data register.

### **33.5.13.3 SPTEF**

SPTEF occurs when the SPI transmit buffer is ready to accept new data. In 8-bit mode, SPTEF is set only after all 8 bits have been moved from SPIx\_DL into the shifter. In 16-bit mode, SPTEF is set only after all 16 bits have been moved from SPIx\_DH:SPIx\_DL into the shifter.

After SPTEF is set, it does not clear until it is serviced. SPTEF has an automatic clearing process that is described in the SPI Status Register details.

### **33.5.13.4 SPMF**

SPMF occurs when the data in the receive data buffer is equal to the data in the SPI Match Register. In 8-bit mode, SPMF is set only after bits 7–0 in the receive data buffer are determined to be equivalent to the value in SPIx\_ML. In 16-bit mode, SPMF is set after bits 15–0 in the receive data buffer are determined to be equivalent to the value in SPIx\_MH:SPIx\_ML.

### **33.5.13.5 TNEAREF**

The TNEAREF bit applies when the FIFO feature is supported.

The TNEAREF flag is set when only one 16-bit word or two 8-bit bytes of data remain in the transmit FIFO provided C3[5] = 0 or when only two 16-bit words or four 8-bit bytes of data remain in the transmit FIFO provided C3[5] = 1. If FIFOMODE is not enabled, ignore this bit.

Clearing this interrupt depends on the state of C3[3] and the status of TNEAREF. Refer to the description of the SPI status (S) register.

### **33.5.13.6 RNFULLF**

The RNFULLF bit applies when the FIFO feature is supported.

RNFULLF is set when more than three 16-bit words or six 8-bit bytes of data remain in the receive FIFO provided C3[4] = 0 or when more than two 16-bit words or four 8-bit bytes of data remain in the receive FIFO provided C3[4] = 1.

Clearing this interrupt depends on the state of C3[3] and the status of RNFULLF. Refer to the description of the SPI status (S) register.

### 33.5.13.7 Asynchronous interrupt in low-power modes

When the CPU is in Wait mode or Stop mode and the SPI module receives a transmission, the SPI module can generate an asynchronous interrupt to wake the CPU from the low power mode. The module generates the asynchronous interrupt only when all of the following conditions apply:

1. C1[SPIE] is set to 1.
2. The CPU is in Wait mode—in which case C2[SPISWAI] must be 1—or in Stop mode where the peripheral bus clock is stopped but internal logic states are retained.
3. The SPI module is in slave mode.
4. The received transmission ends.
5. When the FIFO feature is supported, FIFO mode is disabled: C3[FIFOMODE] is 0.

After the interrupt wakes the CPU and the peripheral bus clock is active again, the SPI module copies the received data from the shifter into the Data register and generates flags or DMA request signals. During the wakeup phase, a continuous transmission from a master would destroy the first received data.

## 33.6 Initialization/application information

This section discusses an example of how to initialize and use the SPI.

### NOTE

When operating the SPI at the maximum baud rate it must be configured for 16 bit operation.

### 33.6.1 Initialization sequence

Before the SPI module can be used for communication, an initialization procedure must be carried out, as follows:

1. Update the Control Register 1 (SPIx\_C1) to enable the SPI and to control interrupt enables. This register also sets the SPI as master or slave, determines clock phase and polarity, and configures the main SPI options.
2. Update the Control Register 2 (SPIx\_C2) to enable additional SPI functions such as the SPI match interrupt feature, the master mode-fault function, and bidirectional mode output as well as to control 8- or 16-bit mode selection and other optional features.
3. Update the Baud Rate Register (SPIx\_BR) to set the prescaler and bit rate divisor for an SPI master.
4. Update the Hardware Match Register (SPIx\_MH:SPIx\_ML) with the value to be compared to the receive data register for triggering an interrupt if hardware match interrupts are enabled.
5. In the master, read SPIx\_S while S[SPTEF] = 1, and then write to the transmit data register (SPIx\_DH:SPIx\_DL) to begin transfer.

### 33.6.2 Pseudo-Code Example

In this example, the SPI module is set up for master mode with only hardware match interrupts enabled. The SPI runs in 16-bit mode at a maximum baud rate of SPI module clock divided by 2. Clock phase and polarity are set for an active-high SPI clock where the first edge on SPSCCK occurs at the start of the first cycle of a data transfer.

SPIx_C1=0x54(%01010100)				
Bit 7	SPIE	=	0	Disables receive and mode fault interrupts
Bit 6	SPE	=	1	Enables the SPI system
Bit 5	SPTIE	=	0	Disables SPI transmit interrupts
Bit 4	MSTR	=	1	Sets the SPI module as a master SPI device
Bit 3	CPOL	=	0	Configures SPI clock as active-high
Bit 2	CPHA	=	1	First edge on SPSCCK at start of first data transfer cycle
Bit 1	SSOE	=	0	Determines SS pin function when mode fault enabled
Bit 0	LSBFE	=	0	SPI serial data transfers start with most significant bit

SPIx_C2 = 0xC0(%11000000)				
Bit 7	SPMIE	=	1	SPI hardware match interrupt enabled
Bit 6	SPIMODE	=	1	Configures SPI for 16-bit mode
Bit 5	TXDMAE	=	0	DMA request disabled
Bit 4	MODFEN	=	0	Disables mode fault function

*Table continues on the next page...*

<b>SPIx_C2 = 0xC0(%11000000)</b>				
Bit 3	BIDIROE	=	0	SPI data I/O pin acts as input
Bit 2	RXDMAE	=	0	DMA request disabled
Bit 1	SPISWAI	=	0	SPI clocks operate in wait mode
Bit 0	SPC0	=	0	uses separate pins for data input and output

<b>SPIx_BR = 0x00(%00000000)</b>				
Bit 7		=	0	Reserved
Bit 6:4		=	000	Sets prescale divisor to 1
Bit 3:0		=	0000	Sets baud rate divisor to 2

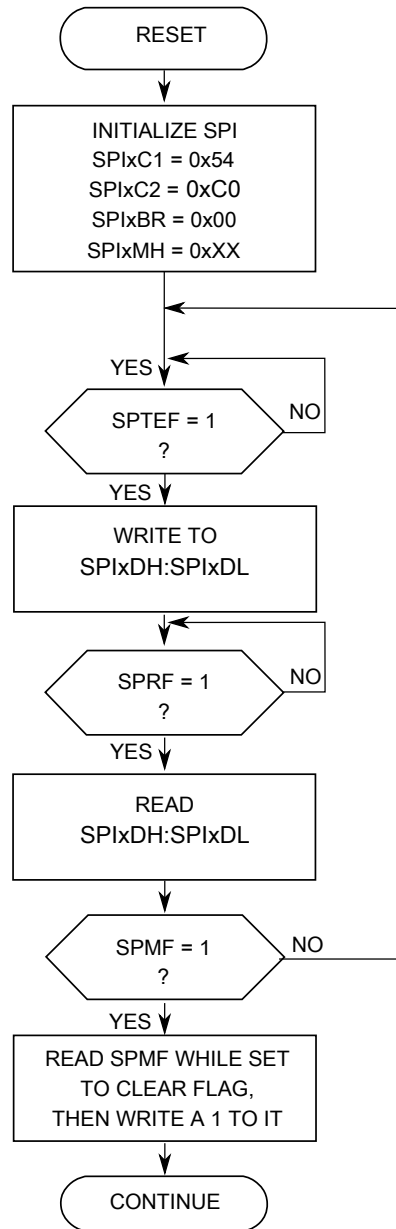
<b>SPIx_S = 0x00(%00000000)</b>				
Bit 7	SPRF	=	0	Flag is set when receive data buffer is full
Bit 6	SPMF	=	0	Flag is set when SPIx_MH/ML = receive data buffer
Bit 5	SPTEF	=	0	Flag is set when transmit data buffer is empty
Bit 4	MODF	=	0	Mode fault flag for master mode
Bit 3:0	RNFULLF, TNEARF, TXFULLF, and RFIFOEf	=	0	Reserved (when FIFOMODE is not present or is 0) or FIFO flags (when FIFOMODE is 1)
Bit 3:0		=	0	FIFOMODE is not enabled

<b>SPIx_MH = 0xXX</b>	
In 16-bit mode, this register holds bits 8–15 of the hardware match buffer. In 8-bit mode, writes to this register will be ignored.	

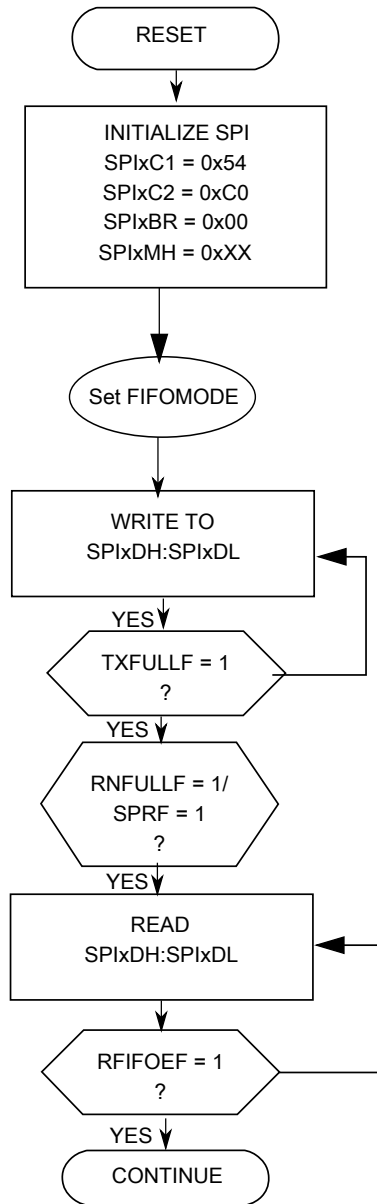
<b>SPIx_ML = 0xXX</b>	
Holds bits 0–7 of the hardware match buffer.	

<b>SPIx_DH = 0xxx</b>	
In 16-bit mode, this register holds bits 8–15 of the data to be transmitted by the transmit buffer and received by the receive buffer.	

<b>SPIx_DL = 0xxx</b>	
Holds bits 0–7 of the data to be transmitted by the transmit buffer and received by the receive buffer.	



**Figure 33-10. Initialization Flowchart Example for SPI Master Device in 16-bit Mode for FIFOMODE = 0**



**Figure 33-11. Initialization Flowchart Example for SPI Master Device in 16-bit Mode for FIFOMODE = 1**





# Chapter 34

## Inter-Integrated Circuit (I2C)

### 34.1 Chip-specific I2C information

#### 34.1.1 I2C instantiation information

This device has two IIC modules. I2Cx are clocked by the system clock so they can support standard IIC communication rates of 100 kbit/s in VLPR mode.

When the package pins associated with IIC have their mux select configured for IIC operation, the pins (SCL and SDA) are driven either by true open drain or in a pseudo open drain configuration. However, only pseudo open drain configuration is for KLx3 family.

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

### 34.2 Introduction

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 at least 400 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*.

## 34.2.1 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
- Double buffering support to achieve higher baud rate

## 34.2.2 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.

## 34.2.3 Block diagram

The following figure is a functional block diagram of the I2C module.

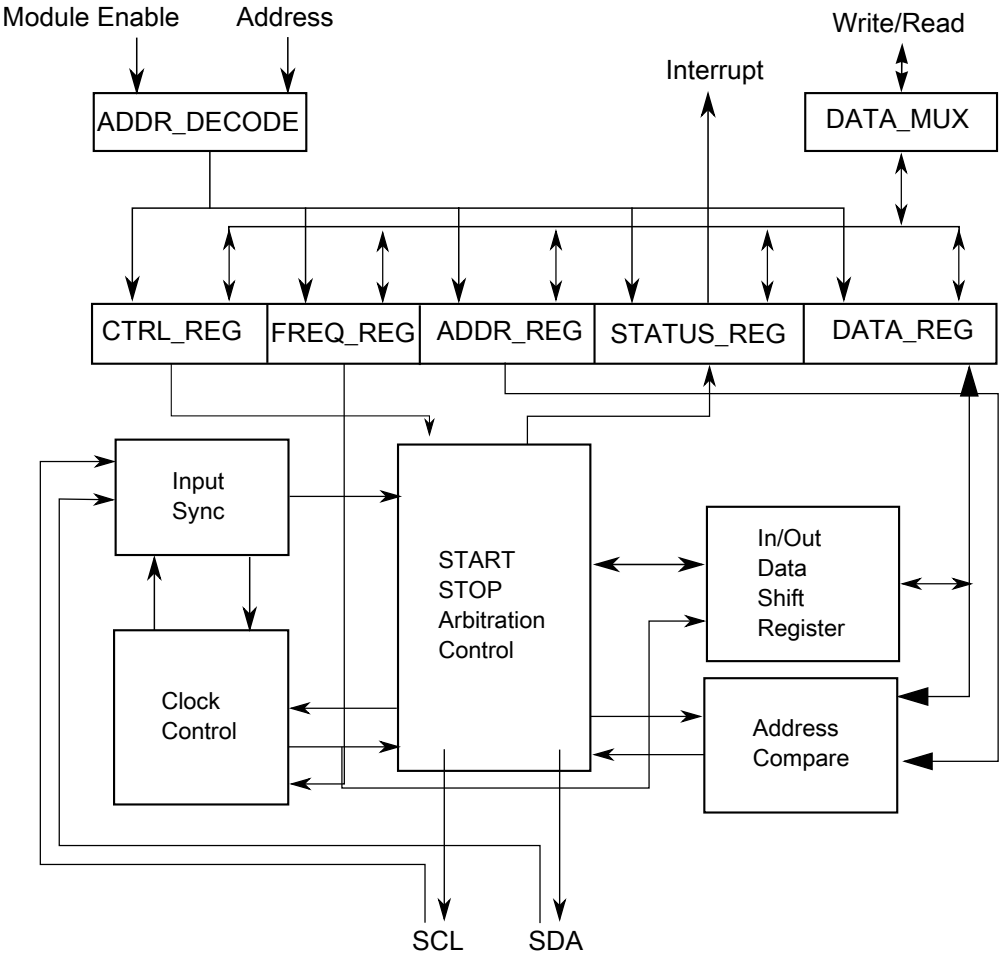


Figure 34-1. I2C Functional block diagram

### 34.3 I<sup>2</sup>C signal descriptions

The signal properties of I<sup>2</sup>C are shown in the table found here.

Table 34-1. 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

## 34.4 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="#">34.4.1/565</a>
4006_6001	I2C Frequency Divider register (I2C0_F)	8	R/W	00h	<a href="#">34.4.2/565</a>
4006_6002	I2C Control Register 1 (I2C0_C1)	8	R/W	00h	<a href="#">34.4.3/566</a>
4006_6003	I2C Status register (I2C0_S)	8	R/W	80h	<a href="#">34.4.4/568</a>
4006_6004	I2C Data I/O register (I2C0_D)	8	R/W	00h	<a href="#">34.4.5/570</a>
4006_6005	I2C Control Register 2 (I2C0_C2)	8	R/W	00h	<a href="#">34.4.6/570</a>
4006_6006	I2C Programmable Input Glitch Filter Register (I2C0_FLT)	8	R/W	00h	<a href="#">34.4.7/571</a>
4006_6007	I2C Range Address register (I2C0_RA)	8	R/W	00h	<a href="#">34.4.8/573</a>
4006_6008	I2C SMBus Control and Status register (I2C0_SMB)	8	R/W	00h	<a href="#">34.4.9/573</a>
4006_6009	I2C Address Register 2 (I2C0_A2)	8	R/W	C2h	<a href="#">34.4.10/575</a>
4006_600A	I2C SCL Low Timeout Register High (I2C0_SLTH)	8	R/W	00h	<a href="#">34.4.11/575</a>
4006_600B	I2C SCL Low Timeout Register Low (I2C0_SLTL)	8	R/W	00h	<a href="#">34.4.12/576</a>
4006_600C	I2C Status register 2 (I2C0_S2)	8	R/W	01h	<a href="#">34.4.13/576</a>
4006_7000	I2C Address Register 1 (I2C1_A1)	8	R/W	00h	<a href="#">34.4.1/565</a>
4006_7001	I2C Frequency Divider register (I2C1_F)	8	R/W	00h	<a href="#">34.4.2/565</a>
4006_7002	I2C Control Register 1 (I2C1_C1)	8	R/W	00h	<a href="#">34.4.3/566</a>
4006_7003	I2C Status register (I2C1_S)	8	R/W	80h	<a href="#">34.4.4/568</a>
4006_7004	I2C Data I/O register (I2C1_D)	8	R/W	00h	<a href="#">34.4.5/570</a>
4006_7005	I2C Control Register 2 (I2C1_C2)	8	R/W	00h	<a href="#">34.4.6/570</a>
4006_7006	I2C Programmable Input Glitch Filter Register (I2C1_FLT)	8	R/W	00h	<a href="#">34.4.7/571</a>
4006_7007	I2C Range Address register (I2C1_RA)	8	R/W	00h	<a href="#">34.4.8/573</a>
4006_7008	I2C SMBus Control and Status register (I2C1_SMB)	8	R/W	00h	<a href="#">34.4.9/573</a>
4006_7009	I2C Address Register 2 (I2C1_A2)	8	R/W	C2h	<a href="#">34.4.10/575</a>
4006_700A	I2C SCL Low Timeout Register High (I2C1_SLTH)	8	R/W	00h	<a href="#">34.4.11/575</a>
4006_700B	I2C SCL Low Timeout Register Low (I2C1_SLTL)	8	R/W	00h	<a href="#">34.4.12/576</a>
4006_700C	I2C Status register 2 (I2C1_S2)	8	R/W	01h	<a href="#">34.4.13/576</a>

### 34.4.1 I2C Address Register 1 (I2Cx\_A1)

This register contains the slave address to be used by the I2C module.

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read	AD[7:1]							0
Write								
Reset	0	0	0	0	0	0	0	0

#### I2Cx\_A1 field descriptions

Field	Description
7–1 AD[7:1]	Address Contains the primary slave address used by the I2C module when it is addressed as a slave. This field is used in the 7-bit address scheme and the lower seven bits in the 10-bit address scheme.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

### 34.4.2 I2C Frequency Divider register (I2Cx\_F)

Address: Base address + 1h offset

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	Multiplier Factor Defines the multiplier factor (mul). This factor is used along with the SCL divider to generate the I2C baud rate.  00 mul = 1 01 mul = 2 10 mul = 4 11 Reserved
ICR	ClockRate Prescales the I2C module clock for bit rate selection. This field and the MULT field determine the I2C baud rate, the SDA hold time, the SCL start hold time, and the SCL stop hold time. For a list of values corresponding to each ICR setting, see <a href="#">I2C divider and hold values</a> .  The SCL divider multiplied by multiplier factor (mul) determines the I2C baud rate.  $\text{I2C baud rate} = \text{I2C module clock speed (Hz)} / (\text{mul} \times \text{SCL divider})$

Table continues on the next page...

### I2Cx\_F field descriptions (continued)

Field	Description																																	
	<p>The SDA hold time is the delay from the falling edge of SCL (I2C clock) to the changing of SDA (I2C data).</p> <p>SDA hold time = I2C module clock period (s) × mul × SDA hold value</p> <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> <p>SCL start hold time = I2C module clock period (s) × mul × SCL start hold value</p> <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> <p>SCL stop hold time = I2C module clock period (s) × mul × SCL stop hold value</p> <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 border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th rowspan="2">MULT</th> <th rowspan="2">ICR</th> <th colspan="3">Hold times (µs)</th> </tr> <tr> <th>SDA</th> <th>SCL Start</th> <th>SCL Stop</th> </tr> </thead> <tbody> <tr> <td>2h</td> <td>00h</td> <td>3.500</td> <td>3.000</td> <td>5.500</td> </tr> <tr> <td>1h</td> <td>07h</td> <td>2.500</td> <td>4.000</td> <td>5.250</td> </tr> <tr> <td>1h</td> <td>0Bh</td> <td>2.250</td> <td>4.000</td> <td>5.250</td> </tr> <tr> <td>0h</td> <td>14h</td> <td>2.125</td> <td>4.250</td> <td>5.125</td> </tr> <tr> <td>0h</td> <td>18h</td> <td>1.125</td> <td>4.750</td> <td>5.125</td> </tr> </tbody> </table>	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
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																														

### 34.4.3 I2C Control Register 1 (I2Cx\_C1)

Address: Base address + 2h offset

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	<p>I2C Enable</p> <p>Enables I2C module operation.</p> <p>0 Disabled 1 Enabled</p>
6 IICIE	<p>I2C Interrupt Enable</p> <p>Enables I2C interrupt requests.</p>

Table continues on the next page...

## I2Cx\_C1 field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
5 MST	Master Mode Select  When MST is changed from 0 to 1, a START signal is generated on the bus and master mode is selected. When this bit changes from 1 to 0, a STOP signal is generated and the mode of operation changes from master to slave.  0 Slave mode 1 Master mode
4 TX	Transmit Mode Select  Selects the direction of master and slave transfers. In master mode this bit must be set according to the type of transfer required. Therefore, for address cycles, this bit is always set. When addressed as a slave this bit must be set by software according to the SRW bit in the status register.  0 Receive 1 Transmit
3 TXAK	Transmit Acknowledge Enable  Specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. The value of SMB[FAACK] affects NACK/ACK generation.  <b>NOTE:</b> SCL is held low until TXAK is written.  0 An acknowledge signal is sent to the bus on the following receiving byte (if FACK is cleared) or the current receiving byte (if FACK is set). 1 No acknowledge signal is sent to the bus on the following receiving data byte (if FACK is cleared) or the current receiving data byte (if FACK is set).
2 RSTA	Repeat START  Writing 1 to this bit generates a repeated START condition provided it is the current master. This bit will always be read as 0. Attempting a repeat at the wrong time results in loss of arbitration.
1 WUEN	Wakeup Enable  The I2C module can wake the MCU from low power mode with no peripheral bus running when slave address matching occurs.  0 Normal operation. No interrupt generated when address matching in low power mode. 1 Enables the wakeup function in low power mode.
0 DMAEN	DMA Enable  Enables or disables the DMA function.  0 All DMA signalling disabled. 1 DMA transfer is enabled. While SMB[FAACK] = 0, the following conditions trigger the DMA request: <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>

*Table continues on the next page...*

### I2Cx\_C1 field descriptions (continued)

Field	Description
	<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>

### 34.4.4 I2C Status register (I2Cx\_S)

Address: Base address + 3h offset

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><b>NOTE:</b> In the buffer mode, TCF is cleared automatically by internal reading or writing the data register I2C_D, with no need waiting for manually reading/writing the I2C data register in Rx/Tx 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>

Table continues on the next page...



## I2Cx\_S field descriptions (continued)

Field	Description
	<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> <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>

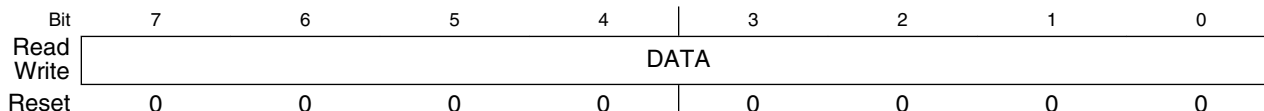
Table continues on the next page...

### I2Cx\_S field descriptions (continued)

Field	Description
0	Acknowledge signal was received after the completion of one byte of data transmission on the bus
1	No acknowledge signal detected

### 34.4.5 I2C Data I/O register (I2Cx\_D)

Address: Base address + 4h offset

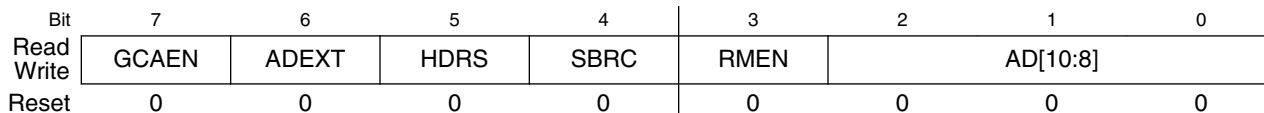


#### I2Cx\_D field descriptions

Field	Description
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>

### 34.4.6 I2C Control Register 2 (I2Cx\_C2)

Address: Base address + 5h offset



## I2Cx\_C2 field descriptions

Field	Description
7 GCAEN	General Call Address Enable Enables general call address. 0 Disabled 1 Enabled
6 ADEXT	Address Extension Controls the number of bits used for the slave address. 0 7-bit address scheme 1 10-bit address scheme
5 HDRS	High Drive Select Controls the drive capability of the I2C pads. 0 Normal drive mode 1 High drive mode
4 SBRC	Slave Baud Rate Control Enables independent slave mode baud rate at maximum frequency, which forces clock stretching on SCL in very fast I2C modes. To a slave, an example of a "very fast" mode is when the master transfers at 40 kbit/s but the slave can capture the master's data at only 10 kbit/s. 0 The slave baud rate follows the master baud rate and clock stretching may occur 1 Slave baud rate is independent of the master baud rate
3 RMEN	Range Address Matching Enable This bit controls the slave address matching for addresses between the values of the A1 and RA registers. When this bit is set, a slave address matching occurs for any address greater than the value of the A1 register and less than or equal to the value of the RA register. 0 Range mode disabled. No address matching occurs for an address within the range of values of the A1 and RA registers. 1 Range mode enabled. Address matching occurs when a slave receives an address within the range of values of the A1 and RA registers.
AD[10:8]	Slave Address Contains the upper three bits of the slave address in the 10-bit address scheme. This field is valid only while the ADEXT bit is set.

## 34.4.7 I2C Programmable Input Glitch Filter Register (I2Cx\_FLT)

Address: Base address + 6h offset

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	<p>Stop Hold Enable</p> <p>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:</p> <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> <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>
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 <math>n</math> I2C module clock cycles, where <math>n=1-15d</math></p>

### 34.4.8 I2C Range Address register (I2Cx\_RA)

Address: Base address + 7h offset

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.

### 34.4.9 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: Base address + 8h offset

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 FACK	<p>Fast NACK/ACK Enable</p> <p>For SMBus packet error checking, the CPU must be able to issue an ACK or NACK according to the result of receiving data byte.</p> <p>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.</p>
6 ALERTEN	<p>SMBus Alert Response Address Enable</p> <p>Enables or disables SMBus alert response address matching.</p> <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>

Table continues on the next page...

**I2Cx\_SMB field descriptions (continued)**

Field	Description
0 SHTF2IE	SHTF2 Interrupt Enable  Enables SCL high and SDA low timeout interrupt.  0 SHTF2 interrupt is disabled 1 SHTF2 interrupt is enabled

**34.4.10 I2C Address Register 2 (I2Cx\_A2)**

Address: Base address + 9h offset

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.

**34.4.11 I2C SCL Low Timeout Register High (I2Cx\_SLTH)**

Address: Base address + Ah offset

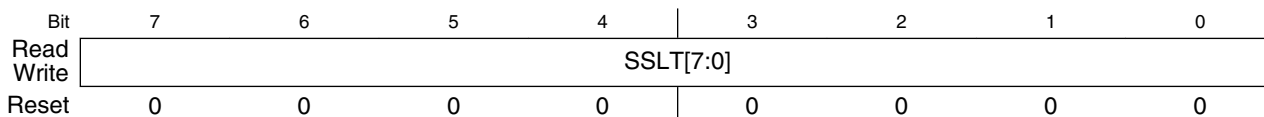
Bit	7	6	5	4	3	2	1	0
Read	SSLT[15:8]							
Write	SSLT[15:8]							
Reset	0	0	0	0	0	0	0	0

**I2Cx\_SLTH field descriptions**

Field	Description
SSLT[15:8]	SSLT[15:8]  Most significant byte of SCL low timeout value that determines the timeout period of SCL low.

### 34.4.12 I2C SCL Low Timeout Register Low (I2Cx\_SLTL)

Address: Base address + Bh offset

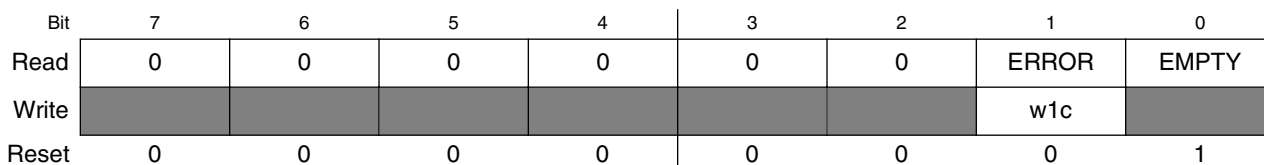


#### I2Cx\_SLTL field descriptions

Field	Description
SSLT[7:0]	SSLT[7:0] Least significant byte of SCL low timeout value that determines the timeout period of SCL low.

### 34.4.13 I2C Status register 2 (I2Cx\_S2)

Address: Base address + Ch offset



#### I2Cx\_S2 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. 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 ERROR	Error flag  Indicates if there are read or write errors with the Tx and Rx buffers.  0 The buffer is not full and all write/read operations have no errors. 1 There are 3 or more write/read errors during the data transfer phase (when the Empty flag is not set and the buffer is busy).

Table continues on the next page...



**I2Cx\_S2 field descriptions (continued)**

Field	Description
0 EMPTY	<p>Empty flag</p> <p>Indicates if the Tx or Rx buffer is empty.</p> <p>0 Tx or Rx buffer is not empty and cannot be written to, that is new data cannot be loaded into the buffer.</p> <p>1 Tx or Rx buffer is empty and can be written to, that is new data can be loaded into the buffer.</p>

## 34.5 Functional description

This section provides a comprehensive functional description of the I2C module.

### 34.5.1 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.

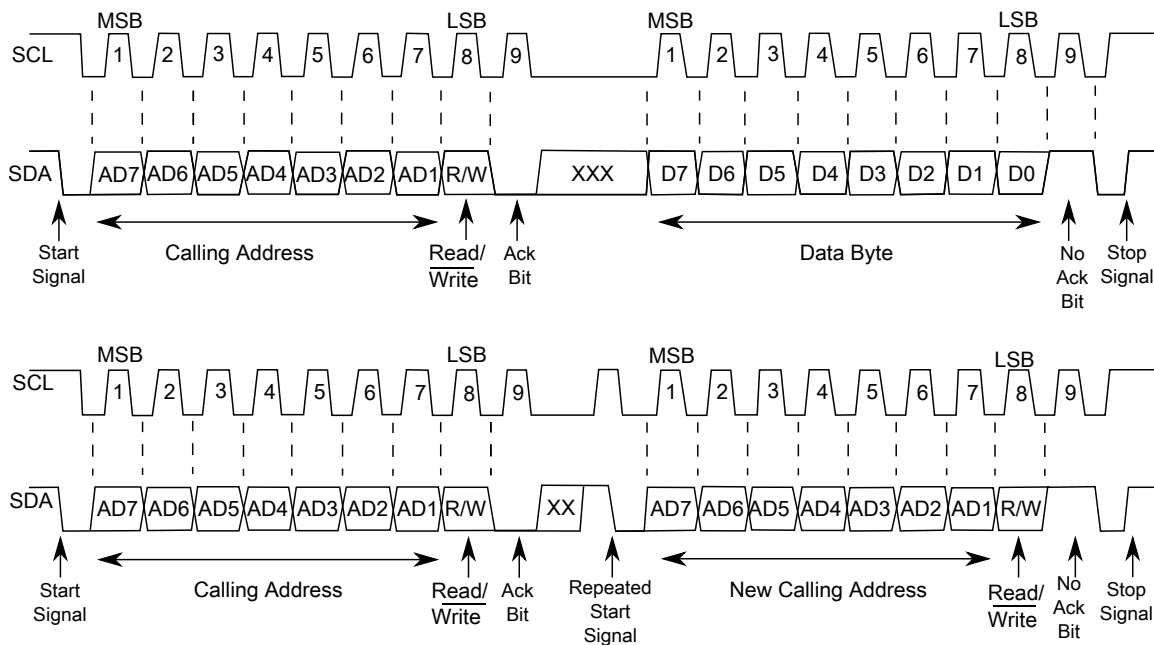


Figure 34-2. I2C bus transmission signals

### 34.5.1.1 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.

### 34.5.1.2 Slave address transmission

Immediately after the START signal, the first byte of a data transfer is the slave address transmitted by the master. This address is a 7-bit calling address followed by an  $R/\overline{W}$  bit. The  $R/\overline{W}$  bit tells the slave the desired direction of data transfer.

- 1 = Read transfer: The slave transmits data to the master
- 0 = Write transfer: The master transmits data to the slave

Only the slave with a calling address that matches the one transmitted by the master responds by sending an acknowledge bit. The slave sends the acknowledge bit by pulling SDA low at the ninth clock.

No two slaves in the system can have the same address. If the I2C module is the master, it must not transmit an address that is equal to its own slave address. The I2C module cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the I2C module reverts to slave mode and operates correctly even if it is being addressed by another master.

### 34.5.1.3 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.

### 34.5.1.4 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.

### 34.5.1.5 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. The master needs to send a NACK signal before sending repeated-START in the buffering mode.

### 34.5.1.6 Arbitration procedure

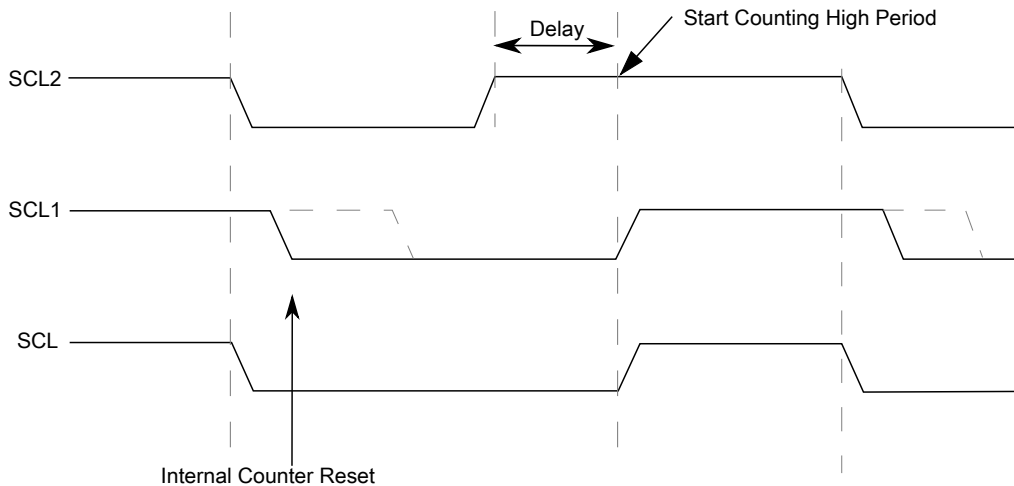
The I2C bus is a true multimaster bus that allows more than one master to be connected on it.

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.

### 34.5.1.7 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 34-3. I2C clock synchronization**

### 34.5.1.8 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.

### 34.5.1.9 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.

### 34.5.1.10 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 34-2. I2C divider and hold values

ICR (hex)	SCL divider	SDA hold value	SCL hold (start) value	SCL hold (stop) value	ICR (hex)	SCL divider (clocks)	SDA hold (clocks)	SCL hold (start) value	SCL hold (stop) value
00	20	7	6	11	20	160	17	78	81
01	22	7	7	12	21	192	17	94	97
02	24	8	8	13	22	224	33	110	113
03	26	8	9	14	23	256	33	126	129
04	28	9	10	15	24	288	49	142	145
05	30	9	11	16	25	320	49	158	161
06	34	10	13	18	26	384	65	190	193
07	40	10	16	21	27	480	65	238	241
08	28	7	10	15	28	320	33	158	161
09	32	7	12	17	29	384	33	190	193
0A	36	9	14	19	2A	448	65	222	225
0B	40	9	16	21	2B	512	65	254	257
0C	44	11	18	23	2C	576	97	286	289
0D	48	11	20	25	2D	640	97	318	321
0E	56	13	24	29	2E	768	129	382	385
0F	68	13	30	35	2F	960	129	478	481
10	48	9	18	25	30	640	65	318	321
11	56	9	22	29	31	768	65	382	385
12	64	13	26	33	32	896	129	446	449
13	72	13	30	37	33	1024	129	510	513
14	80	17	34	41	34	1152	193	574	577
15	88	17	38	45	35	1280	193	638	641
16	104	21	46	53	36	1536	257	766	769
17	128	21	58	65	37	1920	257	958	961
18	80	9	38	41	38	1280	129	638	641
19	96	9	46	49	39	1536	129	766	769
1A	112	17	54	57	3A	1792	257	894	897
1B	128	17	62	65	3B	2048	257	1022	1025
1C	144	25	70	73	3C	2304	385	1150	1153
1D	160	25	78	81	3D	2560	385	1278	1281
1E	192	33	94	97	3E	3072	513	1534	1537
1F	240	33	118	121	3F	3840	513	1918	1921

## 34.5.2 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.

### 34.5.2.1 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 34-3. Master-transmitter addresses slave-receiver with a 10-bit address**

S	Slave address first 7 bits 11110 + AD10 + AD9	R/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.

### 34.5.2.2 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 34-4. Master-receiver addresses a slave-transmitter with a 10-bit address**

S	Slave address first 7 bits 11110 + AD10 + AD9	R/ $\overline{W}$ 0	A1	Slave address second byte AD[8:1]	A2	Sr	Slave address first 7 bits 11110 + AD10 + AD9	R/ $\overline{W}$ 1	A3	Data	A	...	Data	A	P
---	--	------------------------	----	--------------------------------------	----	----	--	------------------------	----	------	---	-----	------	---	---

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter sees an I2C interrupt. User software must ensure that for this interrupt, the contents of the Data register are ignored and not treated as valid data.

### 34.5.3 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.



### 34.5.4 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.

#### 34.5.4.1 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.

##### 34.5.4.1.1 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.

### 34.5.4.1.2 SCL high timeout

When the I2C module has determined that the SMBCLK and SMBDAT signals have been high for at least  $T_{HIGH:MAX}$ , it assumes that the bus is idle.

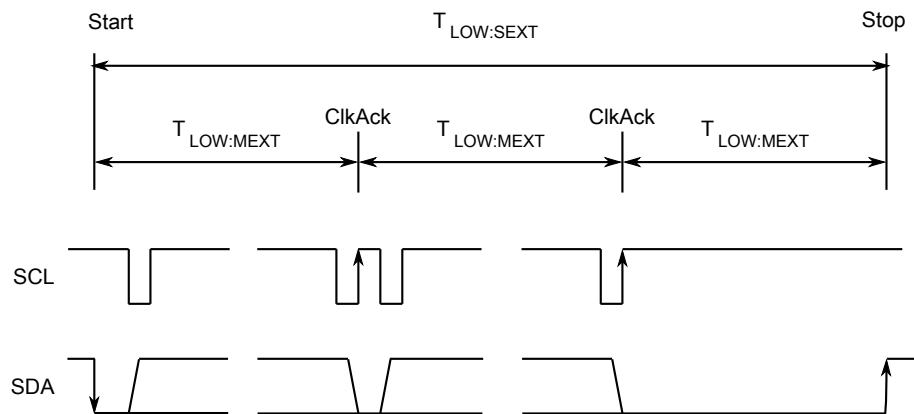
A HIGH timeout occurs after a START condition appears on the bus but before a STOP condition appears on the bus. Any master detecting this scenario can assume the bus is free when either of the following occurs:

- SHTF1 rises.
- The BUSY bit is high and SHTF1 is high.

When the SMBDAT signal is low and the SMBCLK signal is high for a period of time, another kind of timeout occurs. The time period must be defined in software. SHTF2 is used as the flag when the time limit is reached. This flag is also an interrupt resource, so it triggers IICIF.

### 34.5.4.1.3 CSMBCLK TIMEOUT MEXT and CSMBCLK TIMEOUT SEXT

The following figure illustrates the definition of the timeout intervals  $T_{LOW:SEXT}$  and  $T_{LOW:MEXT}$ . When in master mode, the I2C module must not cumulatively extend its clock cycles for a period greater than  $T_{LOW:MEXT}$  within a byte, where each byte is defined as START-to-ACK, ACK-to-ACK, or ACK-to-STOP. When CSMBCLK TIMEOUT MEXT occurs, SMBus MEXT rises and also triggers the SLTF.



**Figure 34-4. Timeout measurement intervals**

A master is allowed to abort the transaction in progress to any slave that violates the  $T_{LOW:SEXT}$  or  $T_{TIMEOUT,MIN}$  specifications. To abort the transaction, the master issues a STOP condition at the conclusion of the byte transfer in progress. When a slave, the I2C module must not cumulatively extend its clock cycles for a period greater than  $T_{LOW:SEXT}$  during any message from the initial START to the STOP. When CSMBCLK TIMEOUT SEXT occurs, SEXT rises and also triggers SLTF.

**NOTE**

CSMBCLK TIMEOUT SEXT and CSMBCLK TIMEOUT MEXT are optional functions that are implemented in the second step.

**34.5.4.2 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.

**34.5.5 Resets**

The I2C module is disabled after a reset. The I2C module cannot cause a core reset.

## 34.5.6 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 34-5. Interrupt summary**

Interrupt source	Status	Flag	Local enable
Complete 1-byte transfer	TCF	IICIF	IICIE
Match of received calling address	IAAS	IICIF	IICIE
Arbitration lost	ARBL	IICIF	IICIE
I <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

### 34.5.6.1 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.

### 34.5.6.2 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.

### 34.5.6.3 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 SSIE bits are both set to 1.

### 34.5.6.4 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.

### 34.5.6.5 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.

### 34.5.6.6 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.

### 34.5.7 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.

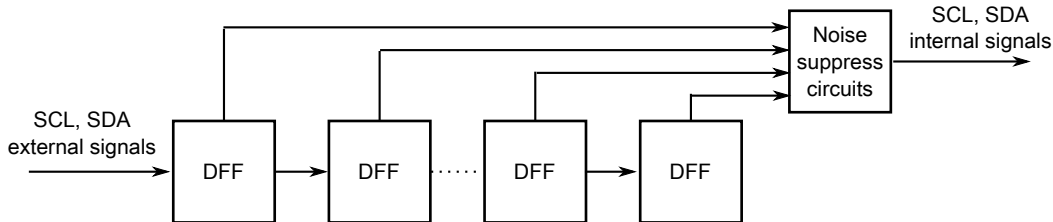


Figure 34-5. Programmable input glitch filter diagram

### 34.5.8 Address matching wake-up

When a primary, range, or general call address match occurs when the I2C module is in slave receive mode, the MCU wakes from a low power mode where no peripheral bus is running.

Data sent on the bus that is the same as a target device address might also wake the target MCU.

After the address matching IAAS bit is set, an interrupt is sent at the end of address matching to wake the core. The IAAS bit must be cleared after the clock recovery.

#### NOTE

After the system recovers and is in Run mode, restart the I2C module if it is needed to transfer packets. To avoid I2C transfer problems resulting from the situation, firmware should prevent the MCU execution of a STOP instruction when the I2C module is in the middle of a transfer unless the Stop mode holdoff feature is used during this period (set FLT[SHEN] to 1).

#### NOTE

After I2C address matching wake-up, the master must wait a time long enough for the slave ISR to finish running and resend start or repeat start signals.

For the SRW bit to function properly, it only supports Address+Write to wake up by I2C address matching. Before entering the next low power mode, Address+Write must be sent to change the SRW status.

### 34.5.9 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.

### 34.5.10 Double buffering mode

In the double buffering mode, the data transfer is processed byte by byte. However, the data can be transferred without waiting for the interrupt or the polling to finish. This means the write/read I2C\_D operation will not block the data transfer, as the hardware has already finished the internal write or read. The benefit is that the baud rate is able to achieve higher speed.

There are several items to consider as follows:

- When initiating a double buffering transfer at Tx side, the user can write 2 values to the I2C\_D buffer before transfer. However, that is allowed only at one time per package frame (due to the buffer depth, and because two-times writes in each ISR are not allowed). The second write to the I2C\_D buffer must wait for the Empty flag. On the other hand, at Rx side the user can read twice in a one-byte transfer (if needed).

#### NOTE

Check Empty flag before write to I2C\_D.

Write twice to the I2C\_D buffer ONLY after the address matching byte. Do not write twice (Address+Data) before START or at the beginning of I2C transfer, especially when the baud rate is very slow.

- To write twice in one frame, during the next-to-last ISR, do a dummy read from the I2C\_D buffer at Tx side (or the TCF will stay high, because the TCF is cleared by write/read operation). In the next-to-last ISR, do not send data again (the buffer data will be under running).
- To keep new ISRs software-compatible with previous ISRs, the write/read I2C\_D operation will not block the internal-hardware-released SCL/SDA signals. At the ACK phase, the bus is released to accept the next byte if the master can send the clock immediately.
- On the slave side, two-times writes to the I2C\_D buffer may be limited by the master's clock and START/repeated-START signal. This is not currently supported, and the master's START/repeated-START signal will break data transfers. To release the bus, do a dummy read or write to the I2C\_D buffer again. It is suggested to send repeated-START/START during intervals as before.
- The master receive should send a NACK in the next-to-last ISR, if it wants to do the STOP or the repeated-START work. The transmitting slave which receives the NACK, will switch to receive mode, and do a dummy read to release SCL and SDA signals.



## 34.6 Initialization/application information

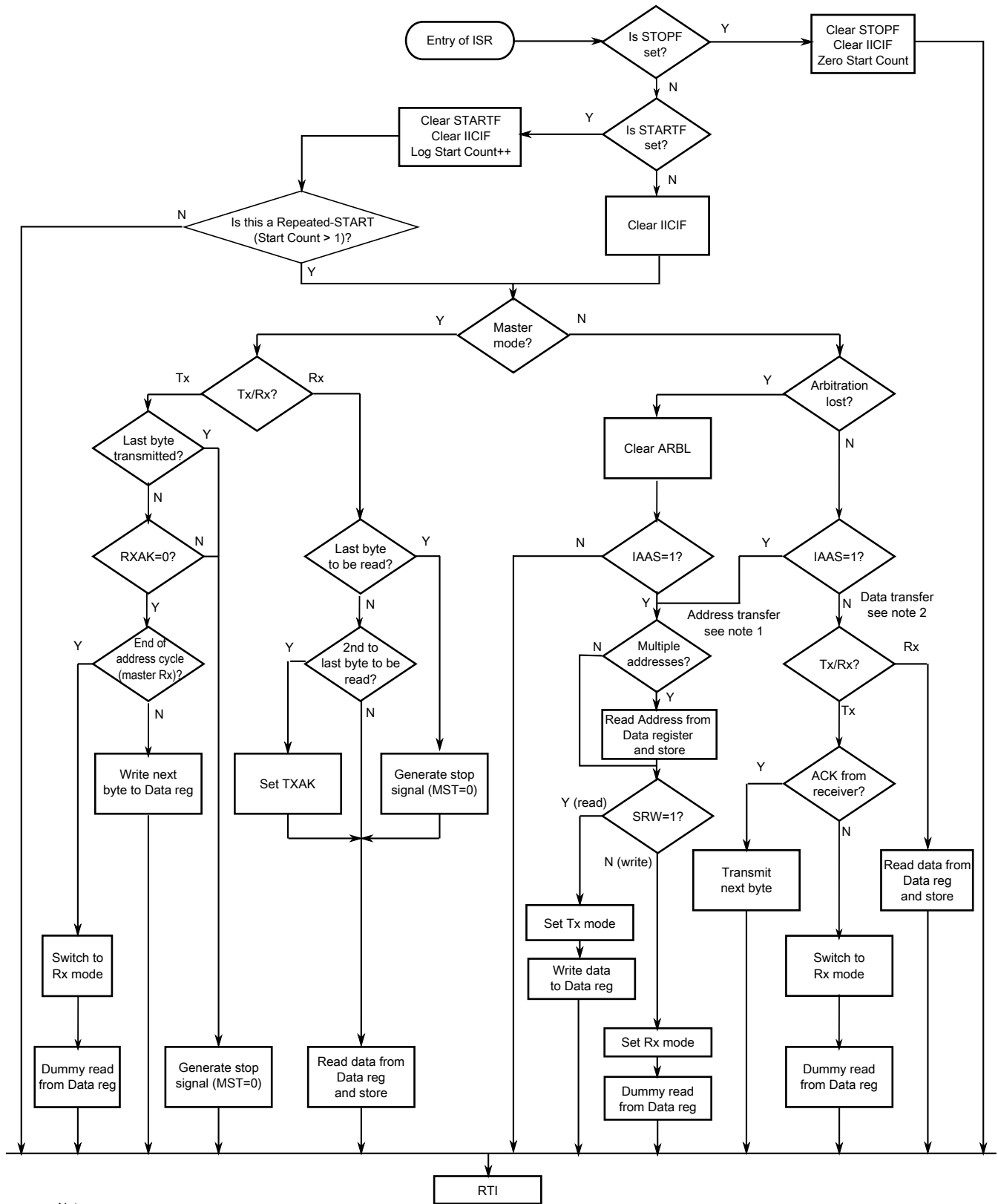
### Module Initialization (Slave)

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)

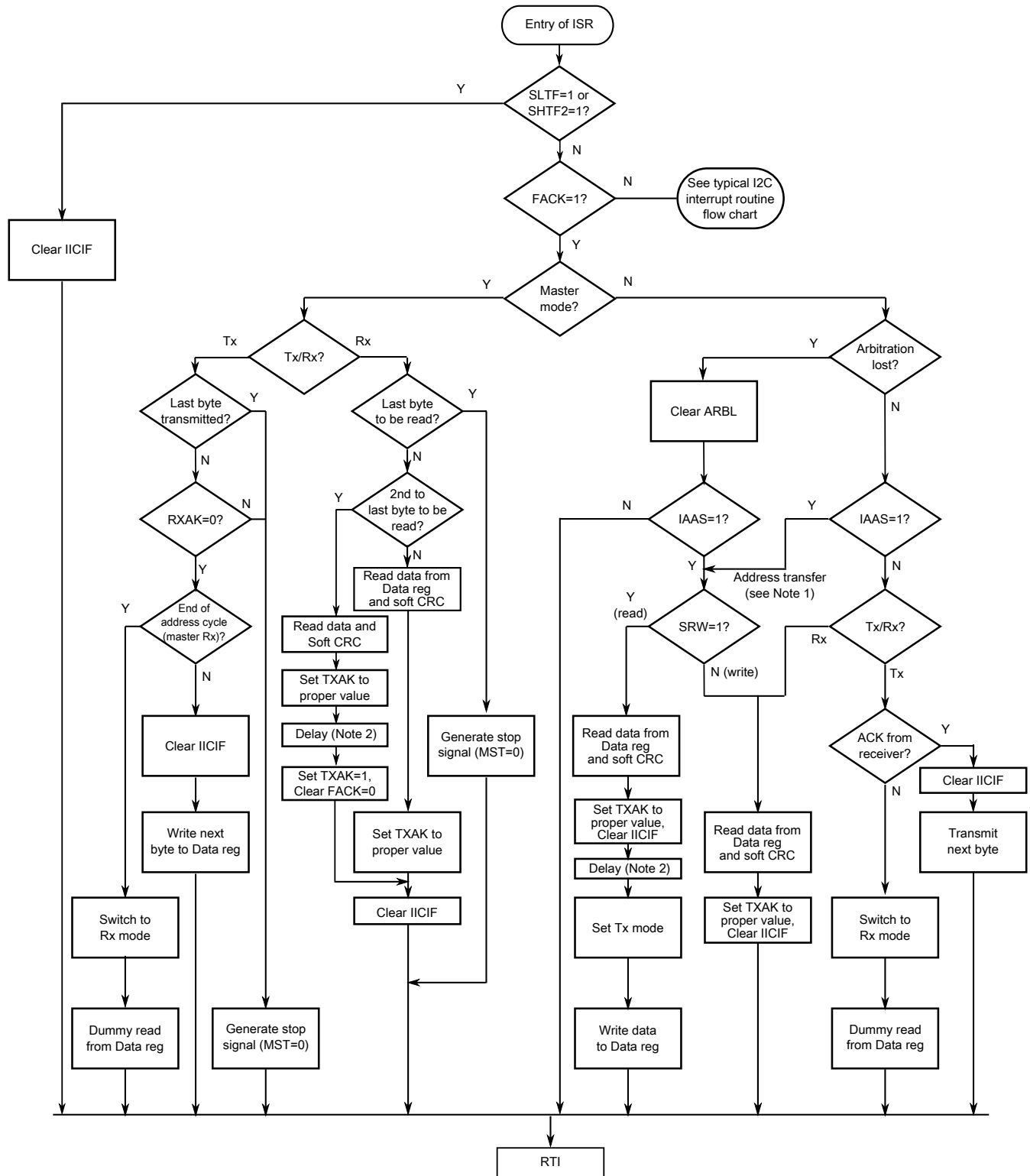
The routine shown in the following figure encompasses both master and slave I2C operations. For slave operation, an incoming I2C message that contains the proper address begins I2C communication. For master operation, communication must be initiated by writing the Data register. An example of an I2C 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 34-6. 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.

Figure 34-7. Typical I2C SMBus interrupt routine



# Chapter 35

## Low Power Universal asynchronous receiver/transmitter (LPUART)

### 35.1 Chip-specific LPUART information

#### 35.1.1 LPUART0 and LPUART1 overview

These modules supports basic UART with DMA interface function, x4 to x32 oversampling of baud-rate.

This module supports LIN slave operation.

The module can remain functional in VLPS mode provided the clock it is using remains enabled.

### 35.2 Introduction

#### 35.2.1 Features

Features of the LPUART module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Programmable baud rates (13-bit modulo divider) with configurable oversampling ratio from 4x to 32x
- Transmit and receive baud rate can operate asynchronous to the bus clock:
  - Baud rate can be configured independently of the bus clock frequency
  - Supports operation in Stop modes
- Interrupt, DMA or polled operation:
  - Transmit data register empty and transmission complete

- Receive data register full
- Receive overrun, parity error, framing error, and noise error
- Idle receiver detect
- Active edge on receive pin
- Break detect supporting LIN
- Receive data match
- Hardware parity generation and checking
- Programmable 8-bit, 9-bit or 10-bit character length
- Programmable 1-bit or 2-bit stop bits
- Three receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
  - Receive data match
- Automatic address matching to reduce ISR overhead:
  - Address mark matching
  - Idle line address matching
  - Address match start, address match end
- Optional 13-bit break character generation / 11-bit break character detection
- Configurable idle length detection supporting 1, 2, 4, 8, 16, 32, 64 or 128 idle characters
- Selectable transmitter output and receiver input polarity

## 35.2.2 Modes of operation

### 35.2.2.1 Stop mode

The LPUART will remain functional during Stop mode, provided the asynchronous transmit and receive clock remains enabled. The LPUART can generate an interrupt or DMA request to cause a wakeup from Stop mode.

### 35.2.2.2 Wait mode

The LPUART can be configured to Stop in Wait modes, when the DOZEEN bit is set. The transmitter and receiver will finish transmitting/receiving the current word.

### 35.2.2.3 Debug mode

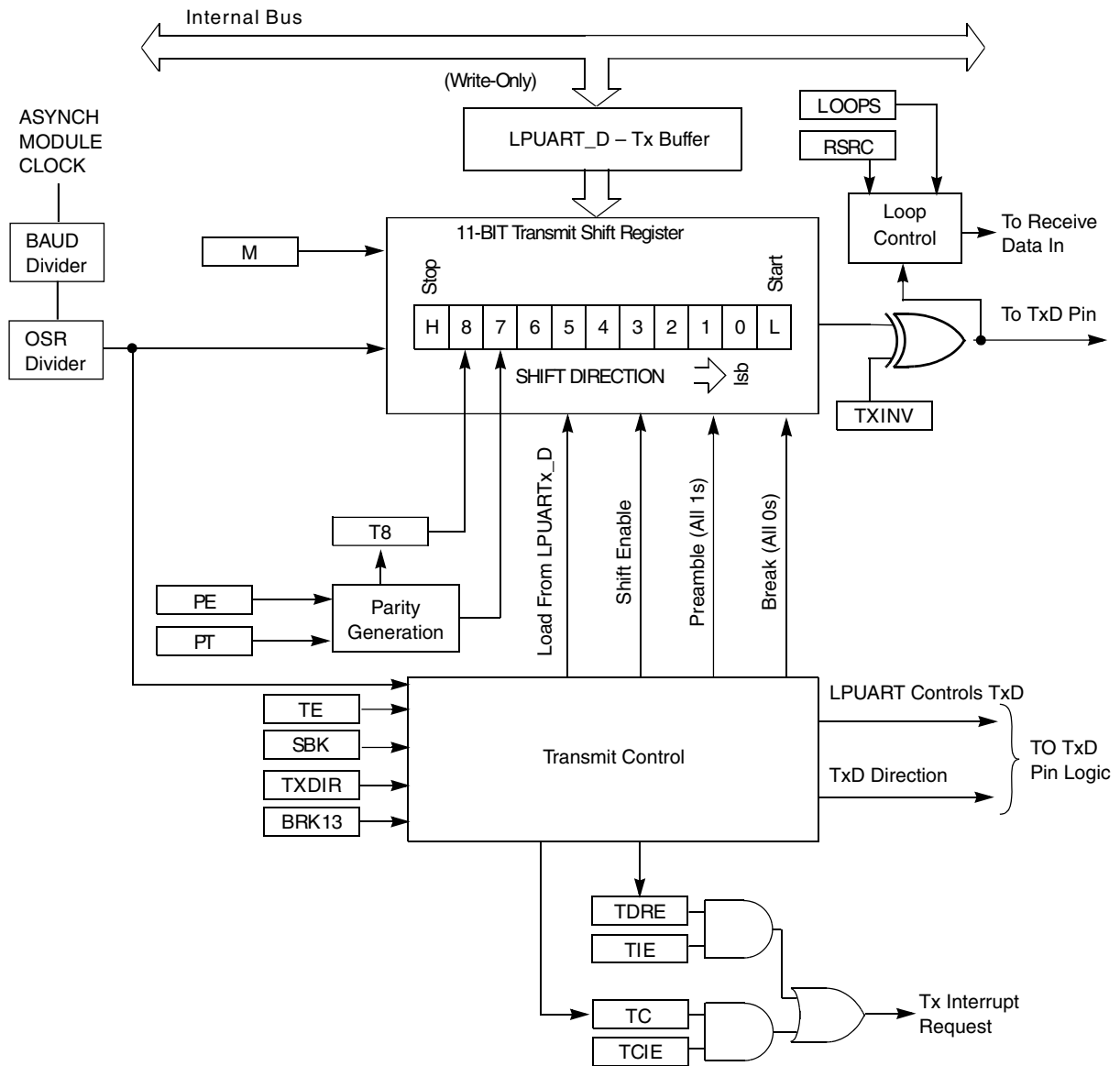
The LPUART remains functional in debug mode.

### 35.2.3 Signal Descriptions

Signal	Description	I/O
LPUART_TX	Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data.	I/O
LPUART_RX	Receive data.	I

### 35.2.4 Block diagram

The following figure shows the transmitter portion of the LPUART.



**Figure 35-1. LPUART transmitter block diagram**

The following figure shows the receiver portion of the LPUART.



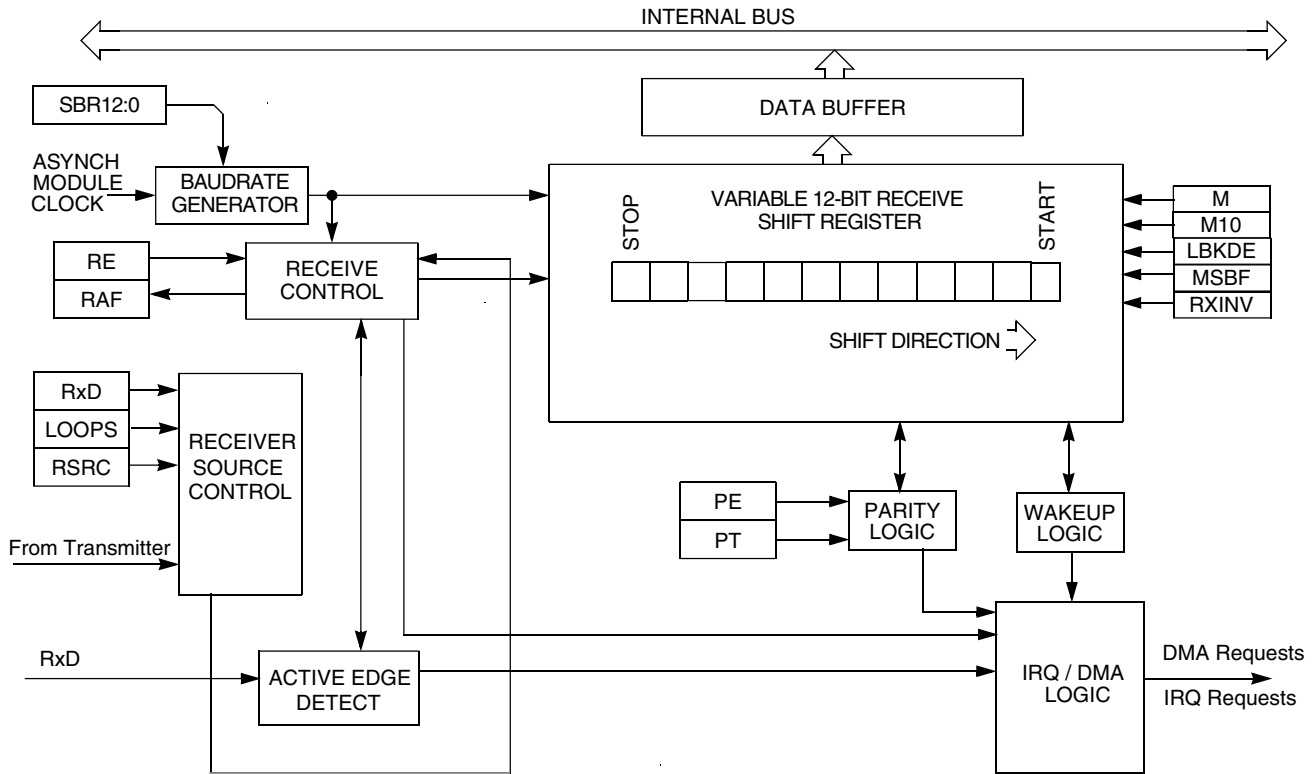


Figure 35-2. LPUART receiver block diagram

### 35.3 Register definition

The LPUART includes registers to control baud rate, select LPUART options, report LPUART status, and for transmit/receive data. Access to an address outside the valid memory map will generate a bus error.

#### LPUART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4005_4000	LPUART Baud Rate Register (LPUART0_BAUD)	32	R/W	0F00_0004h	<a href="#">35.3.1/602</a>
4005_4004	LPUART Status Register (LPUART0_STAT)	32	R/W	00C0_0000h	<a href="#">35.3.2/604</a>
4005_4008	LPUART Control Register (LPUART0_CTRL)	32	R/W	0000_0000h	<a href="#">35.3.3/608</a>
4005_400C	LPUART Data Register (LPUART0_DATA)	32	R/W	0000_1000h	<a href="#">35.3.4/613</a>
4005_4010	LPUART Match Address Register (LPUART0_MATCH)	32	R/W	0000_0000h	<a href="#">35.3.5/615</a>
4005_5000	LPUART Baud Rate Register (LPUART1_BAUD)	32	R/W	0F00_0004h	<a href="#">35.3.1/602</a>
4005_5004	LPUART Status Register (LPUART1_STAT)	32	R/W	00C0_0000h	<a href="#">35.3.2/604</a>
4005_5008	LPUART Control Register (LPUART1_CTRL)	32	R/W	0000_0000h	<a href="#">35.3.3/608</a>
4005_500C	LPUART Data Register (LPUART1_DATA)	32	R/W	0000_1000h	<a href="#">35.3.4/613</a>
4005_5010	LPUART Match Address Register (LPUART1_MATCH)	32	R/W	0000_0000h	<a href="#">35.3.5/615</a>

### 35.3.1 LPUART Baud Rate Register (LPUARTx\_BAUD)

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									0		0					
W	MAEN1	MAEN2	M10	OSR				TDMAE		RDMAE		MATCFG	BOTHEDGE		RESYNCDIS	
Reset	0	0	0	0	1	1	1	1	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	LBKDIE	FXEDGIE	SBNS	SBR												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

#### LPUARTx\_BAUD field descriptions

Field	Description
31 MAEN1	Match Address Mode Enable 1  0 Normal operation. 1 Enables automatic address matching or data matching mode for MATCH[MA1].
30 MAEN2	Match Address Mode Enable 2  0 Normal operation. 1 Enables automatic address matching or data matching mode for MATCH[MA2].
29 M10	10-bit Mode select  The M10 bit causes a tenth bit to be part of the serial transmission. This bit should only be changed when the transmitter and receiver are both disabled.  0 Receiver and transmitter use 8-bit or 9-bit data characters. 1 Receiver and transmitter use 10-bit data characters.
28–24 OSR	Over Sampling Ratio  This field configures the oversampling ratio for the receiver between 4x (00011) and 32x (11111). Writing an invalid oversampling ratio will default to an oversampling ratio of 16 (01111). This field should only be changed when the transmitter and receiver are both disabled.
23 TDMAE	Transmitter DMA Enable  TDMAE configures the transmit data register empty flag, LPUART_STAT[TDRE], to generate a DMA request.  0 DMA request disabled. 1 DMA request enabled.

Table continues on the next page...

## LPUARTx\_BAUD field descriptions (continued)

Field	Description
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 RDMAE	Receiver Full DMA Enable  RDMAE configures the receiver data register full flag, LPUART_STAT[RDRF], to generate a DMA request.  0 DMA request disabled. 1 DMA request enabled.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–18 MATCFG	Match Configuration  Configures the match addressing mode used.  00 Address Match Wakeup 01 Idle Match Wakeup 10 Match On and Match Off 11 Enables RWU on Data Match and Match On/Off
17 BOTHEDGE	Both Edge Sampling  Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given oversampling ratio. This bit must be set for oversampling ratios between x4 and x7 and is optional for higher oversampling ratios. This bit should only be changed when the receiver is disabled.  0 Receiver samples input data using the rising edge of the baud rate clock. 1 Receiver samples input data using the rising and falling edge of the baud rate clock.
16 RESYNCDIS	Resynchronization Disable  When set, disables the resynchronization of the received data word when a data one followed by data zero transition is detected. This bit should only be changed when the receiver is disabled.  0 Resynchronization during received data word is supported 1 Resynchronization during received data word is disabled
15 LBKDIE	LIN Break Detect Interrupt Enable  LBKDIE enables the LIN break detect flag, LBKDIF, to generate interrupt requests.  0 Hardware interrupts from LPUART_STAT[LBKDIF] disabled (use polling). 1 Hardware interrupt requested when LPUART_STAT[LBKDIF] flag is 1.
14 RXEDGIE	RX Input Active Edge Interrupt Enable  Enables the receive input active edge, RXEDGIF, to generate interrupt requests. Changing CTRL[LOOP] or CTRL[RSRC] when RXEDGIE is set can cause the RXEDGIF to set.  0 Hardware interrupts from LPUART_STAT[RXEDGIF] disabled (use polling). 1 Hardware interrupt requested when LPUART_STAT[RXEDGIF] flag is 1.
13 SBNS	Stop Bit Number Select  SBNS determines whether data characters are one or two stop bits. This bit should only be changed when the transmitter and receiver are both disabled.

*Table continues on the next page...*

**LPUARTx\_BAUD field descriptions (continued)**

Field	Description
	0 One stop bit. 1 Two stop bits.
SBR	Baud Rate Modulo Divisor.  The 13 bits in SBR[12:0] set the modulo divide rate for the baud rate generator. When SBR is 1 - 8191, the baud rate equals "baud clock / ((OSR+1) × SBR)". The 13-bit baud rate setting [SBR12:SBR0] must only be updated when the transmitter and receiver are both disabled (LPUART_CTRL[RE] and LPUART_CTRL[TE] are both 0).

**35.3.2 LPUART Status Register (LPUARTx\_STAT)**

Address: Base address + 4h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LBKDIF	FXEDGIF	MSBF	RXINV	RWUID	BRK13	LBKDE	RAF	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W	w1c	w1c										w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MA1F	MA2F	0													
W	w1c	w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**LPUARTx\_STAT field descriptions**

Field	Description
31 LBKDIF	LIN Break Detect Interrupt Flag  LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it.

*Table continues on the next page...*

LPUARTx\_STAT field descriptions (continued)

Field	Description
	<p>0 No LIN break character has been detected.</p> <p>1 LIN break character has been detected.</p>
30 RXEDGIF	<p>LPUART_RX Pin Active Edge Interrupt Flag</p> <p>RXEDGIF is set when an active edge, falling if RXINV = 0, rising if RXINV=1, on the LPUART_RX pin occurs. RXEDGIF is cleared by writing a 1 to it.</p> <p>0 No active edge on the receive pin has occurred.</p> <p>1 An active edge on the receive pin has occurred.</p>
29 MSBF	<p>MSB First</p> <p>Setting this bit reverses the order of the bits that are transmitted and received on the wire. This bit does not affect the polarity of the bits, the location of the parity bit or the location of the start or stop bits. This bit should only be changed when the transmitter and receiver are both disabled.</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.</p> <p>1 MSB (bit9, bit8, bit7 or bit6) is the first bit that is transmitted following the start bit depending on the setting of CTRL[M], CTRL[PE] and BAUD[M10]. Further, the first bit received after the start bit is identified as bit9, bit8, bit7 or bit6 depending on the setting of CTRL[M] and CTRL[PE].</p>
28 RXINV	<p>Receive Data Inversion</p> <p>Setting this bit reverses the polarity of the received data input.</p> <p><b>NOTE:</b> Setting RXINV inverts the LPUART_RX input for all cases: data bits, start and stop bits, break, and idle.</p> <p>0 Receive data not inverted.</p> <p>1 Receive data inverted.</p>
27 RWUID	<p>Receive Wake Up Idle Detect</p> <p>For RWU on idle character, RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. For address match wakeup, RWUID controls if the IDLE bit is set when the address does not match. This bit should only be changed when the receiver is disabled.</p> <p>0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. During address match wakeup, the IDLE bit does not get set when an address does not match.</p> <p>1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. During address match wakeup, the IDLE bit does get set when an address does not match.</p>
26 BRK13	<p>Break Character Generation Length</p> <p>BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. This bit should only be changed when the transmitter is disabled.</p> <p>0 Break character is transmitted with length of 10 bit times (if M = 0, SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1).</p> <p>1 Break character is transmitted with length of 13 bit times (if M = 0, SBNS = 0) or 14 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 15 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 16 (if M10 = 1, SNBS = 1).</p>
25 LBKDE	<p>LIN Break Detection Enable</p> <p>LBKDE selects a longer break character detection length. While LBKDE is set, receive data is not stored in the receive data buffer.</p>

Table continues on the next page...

## LPUARTx\_STAT field descriptions (continued)

Field	Description
	<p>0 Break character is detected at length 10 bit times (if M = 0, SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1).</p> <p>1 Break character is detected at length of 11 bit times (if M = 0, SBNS = 0) or 12 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 14 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 15 (if M10 = 1, SNBS = 1).</p>
24 RAF	<p>Receiver Active Flag</p> <p>RAF is set when the receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line.</p> <p>0 LPUART receiver idle waiting for a start bit. 1 LPUART receiver active (LPUART_RX input not idle).</p>
23 TDRE	<p>Transmit Data Register Empty Flag</p> <p>TDRE will set when the transmit data register (LPUART_DATA) is empty. To clear TDRE, write to the LPUART data register (LPUART_DATA).</p> <p>TDRE is not affected by a character that is in the process of being transmitted, it is updated at the start of each transmitted character.</p> <p>0 Transmit data buffer full. 1 Transmit data buffer empty.</p>
22 TC	<p>Transmission Complete Flag</p> <p>TC is cleared when there is a transmission in progress or when a preamble or break character is loaded. 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 writing to LPUART_DATA to transmit new data, queuing a preamble by clearing and then setting LPUART_CTRL[TE], queuing a break character by writing 1 to LPUART_CTRL[SBK].</p> <p>0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete).</p>
21 RDRF	<p>Receive Data Register Full Flag</p> <p>RDRF is set when the receive buffer (LPUART_DATA) is full. To clear RDRF, read the LPUART_DATA register.</p> <p>A character that is in the process of being received does not cause a change in RDRF until the entire character is received. Even if RDRF is set, the character will continue to be received until an overrun condition occurs once the entire character is received.</p> <p>0 Receive data buffer empty. 1 Receive data buffer full.</p>
20 IDLE	<p>Idle Line Flag</p> <p>IDLE is set when the LPUART receive line becomes idle for a full character time after a period of activity. When ILT is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bits time count toward the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.</p> <p>To clear IDLE, write logic 1 to the IDLE flag. After IDLE has been cleared, it cannot become set again until after a new character has been stored in the receive buffer or a LIN break character has set the LBKDIF flag. IDLE is set only once even if the receive line remains idle for an extended period.</p>

Table continues on the next page...

## LPUARTx\_STAT field descriptions (continued)

Field	Description
	0 No idle line detected. 1 Idle line was detected.
19 OR	<b>Receiver Overrun Flag</b>  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 LPUART data registers is not affected. If LBKDE is enabled and a LIN Break is detected, the OR field asserts if LBKDIF is not cleared before the next data character is received.  While the OR flag is set, no additional data is stored in the data buffer even if sufficient room exists. To clear OR, write logic 1 to the OR flag.  0 No overrun. 1 Receive overrun (new LPUART data lost).
18 NF	<b>Noise Flag</b>  The advanced sampling technique used in the receiver takes three samples in each of the received bits. If any of these samples disagrees with the rest of the samples within any bit time in the frame then noise is detected for that character. NF is set whenever the next character to be read from LPUART_DATA was received with noise detected within the character. To clear NF, write logic one to the NF.  0 No noise detected. 1 Noise detected in the received character in LPUART_DATA.
17 FE	<b>Framing Error Flag</b>  FE is set whenever the next character to be read from LPUART_DATA was received with logic 0 detected where a stop bit was expected. To clear NF, write logic one to the NF.  0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
16 PF	<b>Parity Error Flag</b>  PF is set whenever the next character to be read from LPUART_DATA was received when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, write a logic one to the PF.  0 No parity error. 1 Parity error.
15 MA1F	<b>Match 1 Flag</b>  MA1F is set whenever the next character to be read from LPUART_DATA matches MA1. To clear MA1F, write a logic one to the MA1F.  0 Received data is not equal to MA1 1 Received data is equal to MA1
14 MA2F	<b>Match 2 Flag</b>  MA2F is set whenever the next character to be read from LPUART_DATA matches MA2. To clear MA2F, write a logic one to the MA2F.  0 Received data is not equal to MA2 1 Received data is equal to MA2

Table continues on the next page...

**LPUARTx\_STAT field descriptions (continued)**

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

**35.3.3 LPUART Control Register (LPUARTx\_CTRL)**

This read/write register controls various optional features of the LPUART system. This register should only be altered when the transmitter and receiver are both disabled.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
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			IDLECFG			LOOPS	DOZEEN	RSR C	M	WAKE	ILT	PE	PT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**LPUARTx\_CTRL field descriptions**

Field	Description
31 R8T9	Receive Bit 8 / Transmit Bit 9  R8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When reading 9-bit or 10-bit data, read R8 before reading LPUART_DATA.  T9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When writing 10-bit data, write T9 before writing LPUART_DATA. If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.
30 R9T8	Receive Bit 9 / Transmit Bit 8  R9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When reading 10-bit data, read R9 before reading LPUART_DATA  T8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When writing 9-bit or 10-bit data, write T8 before writing LPUART_DATA. If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, they it need not be written each time LPUART_DATA is written.

Table continues on the next page...



## LPUARTx\_CTRL field descriptions (continued)

Field	Description
29 TXDIR	<p>LPUART_TX Pin Direction in Single-Wire Mode</p> <p>When the LPUART is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the LPUART_TX pin. When clearing TXDIR, the transmitter will finish receiving the current character (if any) before the receiver starts receiving data from the LPUART_TX pin.</p> <p>0 LPUART_TX pin is an input in single-wire mode. 1 LPUART_TX pin is an output in single-wire mode.</p>
28 TXINV	<p>Transmit Data Inversion</p> <p>Setting this bit reverses the polarity of the transmitted data output.</p> <p><b>NOTE:</b> Setting TXINV inverts the LPUART_TX output for all cases: data bits, start and stop bits, break, and idle.</p> <p>0 Transmit data not inverted. 1 Transmit data inverted.</p>
27 ORIE	<p>Overrun Interrupt Enable</p> <p>This bit enables the overrun flag (OR) to generate hardware interrupt requests.</p> <p>0 OR interrupts disabled; use polling. 1 Hardware interrupt requested when OR is set.</p>
26 NEIE	<p>Noise Error Interrupt Enable</p> <p>This bit enables the noise flag (NF) to generate hardware interrupt requests.</p> <p>0 NF interrupts disabled; use polling. 1 Hardware interrupt requested when NF is set.</p>
25 FEIE	<p>Framing Error Interrupt Enable</p> <p>This bit enables the framing error flag (FE) to generate hardware interrupt requests.</p> <p>0 FE interrupts disabled; use polling. 1 Hardware interrupt requested when FE is set.</p>
24 PEIE	<p>Parity Error Interrupt Enable</p> <p>This bit enables the parity error flag (PF) to generate hardware interrupt requests.</p> <p>0 PF interrupts disabled; use polling. 1 Hardware interrupt requested when PF is set.</p>
23 TIE	<p>Transmit Interrupt Enable</p> <p>Enables STAT[TDRE] to generate interrupt requests.</p> <p>0 Hardware interrupts from TDRE disabled; use polling. 1 Hardware interrupt requested when TDRE flag is 1.</p>
22 TCIE	<p>Transmission Complete Interrupt Enable for</p> <p>TCIE enables the transmission complete flag, TC, to generate interrupt requests.</p> <p>0 Hardware interrupts from TC disabled; use polling. 1 Hardware interrupt requested when TC flag is 1.</p>

Table continues on the next page...

## LPUARTx\_CTRL field descriptions (continued)

Field	Description
21 RIE	<p>Receiver Interrupt Enable</p> <p>Enables STAT[RDRF] to generate interrupt requests.</p> <p>0 Hardware interrupts from RDRF disabled; use polling. 1 Hardware interrupt requested when RDRF flag is 1.</p>
20 ILIE	<p>Idle Line Interrupt Enable</p> <p>ILIE enables the idle line flag, STAT[IDLE], to generate interrupt requests.</p> <p>0 Hardware interrupts from IDLE disabled; use polling. 1 Hardware interrupt requested when IDLE flag is 1.</p>
19 TE	<p>Transmitter Enable</p> <p>Enables the LPUART transmitter. TE can also be used to queue an idle preamble by clearing and then setting TE. When TE is cleared, this register bit will read as 1 until the transmitter has completed the current character and the LPUART_TX pin is tristated.</p> <p>0 Transmitter disabled. 1 Transmitter enabled.</p>
18 RE	<p>Receiver Enable</p> <p>Enables the LPUART receiver. When RE is written to 0, this register bit will read as 1 until the receiver finishes receiving the current character (if any).</p> <p>0 Receiver disabled. 1 Receiver enabled.</p>
17 RWU	<p>Receiver Wakeup Control</p> <p>This field can be set to place the LPUART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when CTRL[WAKE] is clear or an address match when CTRL[WAKE] is set with STAT[RWUID] is clear.</p> <p><b>NOTE:</b> RWU must be set only with CTRL[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by STAT[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the LPUART 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 be reasserted.</p> <p>0 Normal receiver operation. 1 LPUART receiver in standby waiting for wakeup condition.</p>
16 SBK	<p>Send Break</p> <p>Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 to 13, or 13 to 16 if LPUART_STATBRK13 is set, bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK.</p> <p>0 Normal transmitter operation. 1 Queue break character(s) to be sent.</p>
15 MA1IE	<p>Match 1 Interrupt Enable</p>

Table continues on the next page...

## LPUARTx\_CTRL field descriptions (continued)

Field	Description
	0 MA1F interrupt disabled 1 MA1F interrupt enabled
14 MA2IE	Match 2 Interrupt Enable 0 MA2F interrupt disabled 1 MA2F interrupt enabled
13–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 IDLECFG	Idle Configuration Configures the number of idle characters that must be received before the IDLE flag is set.  000 1 idle character 001 2 idle characters 010 4 idle characters 011 8 idle characters 100 16 idle characters 101 32 idle characters 110 64 idle characters 111 128 idle characters
7 LOOPS	Loop Mode Select  When LOOPS is set, the LPUART_RX pin is disconnected from the LPUART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function.  0 Normal operation - LPUART_RX and LPUART_TX use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input (see RSRC bit).
6 DOZEEN	Doze Enable  0 LPUART is enabled in Doze mode. 1 LPUART is disabled in Doze 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 Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the LPUART does not use the LPUART_RX pin. 1 Single-wire LPUART mode where the LPUART_TX pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select  0 Receiver and transmitter use 8-bit data characters. 1 Receiver and transmitter use 9-bit data characters.
3 WAKE	Receiver Wakeup Method Select  Determines which condition wakes the LPUART when RWU=1: <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>

*Table continues on the next page...*

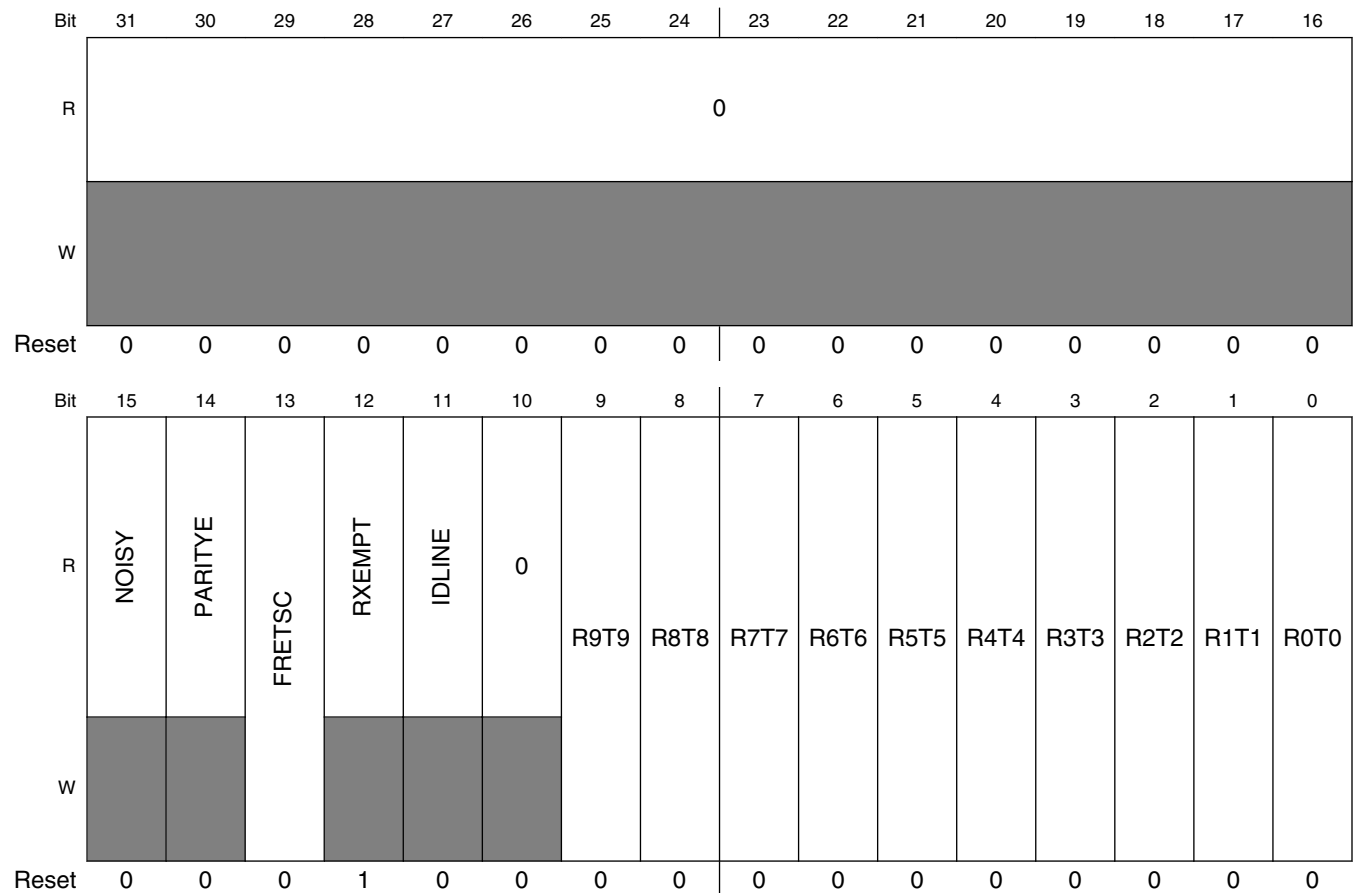
## LPUARTx\_CTRL field descriptions (continued)

Field	Description
	0 Configures RWU for idle-line wakeup. 1 Configures RWU with 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> In case the LPUART is programmed with ILT = 1, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count.  0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.
1 PE	Parity Enable  Enables hardware parity generation and checking. When parity is enabled, the bit immediately before the stop bit is treated as the parity bit.  0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	Parity Type  Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even.  0 Even parity. 1 Odd parity.

### 35.3.4 LPUART Data Register (LPUARTx\_DATA)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status flags.

Address: Base address + Ch offset



**LPUARTx\_DATA field descriptions**

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 NOISY	The current received dataword contained in DATA[R9:R0] was received with noise. 0 The dataword was received without noise. 1 The data was received with noise.
14 PARITYE	The current received dataword contained in DATA[R9:R0] was received with a parity error.

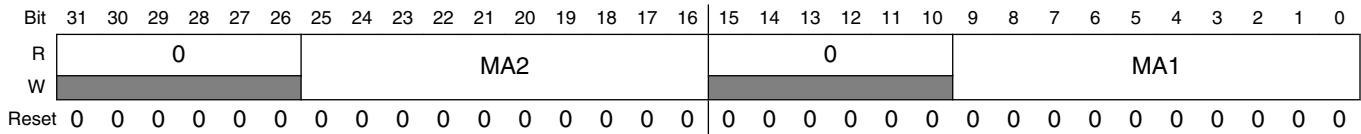
Table continues on the next page...

## LPUARTx\_DATA field descriptions (continued)

Field	Description
	0 The dataword was received without a parity error. 1 The dataword was received with a parity error.
13 FRETSC	Frame Error / Transmit Special Character  For reads, indicates the current received dataword contained in DATA[R9:R0] was received with a frame error. For writes, indicates a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 is used to indicate a break character when 0 and a idle character when 1, he contents of DATA[T8:T0] should be zero.  0 The dataword was received without a frame error on read, transmit a normal character on write. 1 The dataword was received with a frame error, transmit an idle or break character on transmit.
12 RXEMPT	Receive Buffer Empty  Asserts when there is no data in the receive buffer. This field does not take into account data that is in the receive shift register.  0 Receive buffer contains valid data. 1 Receive buffer is empty, data returned on read is not valid.
11 IDLIN	Idle Line  Indicates the receiver line was idle before receiving the character in DATA[9:0]. Unlike the IDLE flag, this bit can set for the first character received when the receiver is first enabled.  0 Receiver was not idle before receiving this character. 1 Receiver was idle before receiving this character.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 R9T9	Read receive data buffer 9 or write transmit data buffer 9.
8 R8T8	Read receive data buffer 8 or write transmit data buffer 8.
7 R7T7	Read receive data buffer 7 or write transmit data buffer 7.
6 R6T6	Read receive data buffer 6 or write transmit data buffer 6.
5 R5T5	Read receive data buffer 5 or write transmit data buffer 5.
4 R4T4	Read receive data buffer 4 or write transmit data buffer 4.
3 R3T3	Read receive data buffer 3 or write transmit data buffer 3.
2 R2T2	Read receive data buffer 2 or write transmit data buffer 2.
1 R1T1	Read receive data buffer 1 or write transmit data buffer 1.
0 R0T0	Read receive data buffer 0 or write transmit data buffer 0.

### 35.3.5 LPUART Match Address Register (LPUARTx\_MATCH)

Address: Base address + 10h offset



#### LPUARTx\_MATCH field descriptions

Field	Description
31–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–16 MA2	Match Address 2  The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear.
15–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MA1	Match Address 1  The MA1 and MA2 registers are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the data register. If a match fails, the following data is discarded. Software should only write a MA register when the associated BAUD[MAEN] bit is clear.

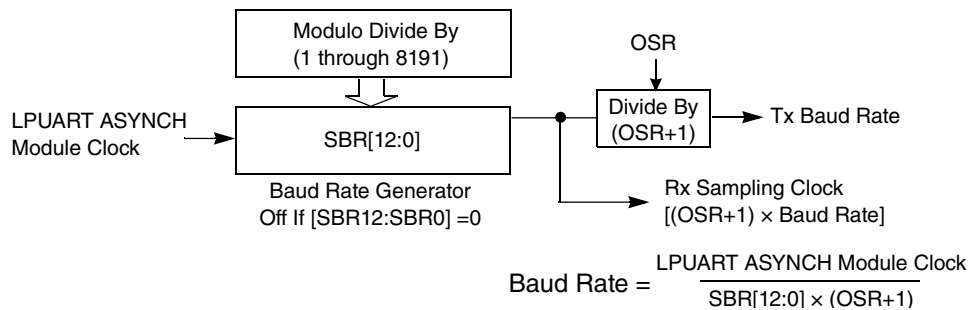
## 35.4 Functional description

The LPUART supports full-duplex, asynchronous, NRZ serial communication and comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. The following describes each of the blocks of the LPUART.

### 35.4.1 Baud rate generation

A 13-bit modulus 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 baud clock divisor for the asynchronous LPUART baud clock. The SBR bits are in the LPUART baud rate registers, BDH and BDL. The baud rate clock drives the receiver,

while the transmitter is driven by the baud rate clock divided by the over sampling ratio. Depending on the over sampling ratio, the receiver has an acquisition rate of 4 to 32 samples per bit time.



**Figure 35-3. LPUART baud rate generation**

Baud rate generation is subject to two sources of error:

- Integer division of the asynchronous LPUART baud clock may not give the exact target frequency.
- Synchronization with the asynchronous LPUART baud clock can cause phase shift.

### 35.4.2 Transmitter functional description

This section describes the overall block diagram for the LPUART transmitter, as well as specialized functions for sending break and idle characters.

The transmitter output (LPUART\_TX) idle state defaults to logic high, CTRL[TXINV] is cleared following reset. The transmitter output is inverted by setting CTRL[TXINV]. The transmitter is enabled by setting the CTRL[TE] bit. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the LPUART data register.

The central element of the LPUART transmitter is the transmit shift register that is 10-bit to 13 bits long depending on the setting in the CTRL[M], BAUD[M10] and BAUD[SBNS] control bits. For the remainder of this section, assume CTRL[M], BAUD[M10] and BAUD[SBNS] are cleared, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new character, the value waiting in the transmit data register is transferred to the shift register, synchronized with the baud rate clock, and the transmit data register empty (STAT[TDRE]) status flag is set to indicate another character may be written to the transmit data buffer at LPUART\_DATA.



If no new character is waiting in the transmit data buffer after a stop bit is shifted out the LPUART\_TX pin, the transmitter sets the transmit complete flag and enters an idle mode, with LPUART\_TX high, waiting for more characters to transmit.

Writing 0 to CTRL[TE] does not immediately disable the transmitter. The current transmit activity in progress must first be completed (that could include a data character, idle character or break character), although the transmitter will not start transmitting another character.

### 35.4.2.1 Send break and queued idle

The LPUART\_CTRL[SBK] bit sends break characters originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0, 10-bit to 12-bit times including the start and stop bits. A longer break of 13-bit times can be enabled by setting LPUART\_STAT[BRK13]. Normally, a program would wait for LPUART\_STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, write 1, and then write 0 to the LPUART\_CTRL[SBK] bit. This action queues a break character to be sent as soon as the shifter is available. If LPUART\_CTRL[SBK] remains 1 when the queued break moves into the shifter, synchronized to the baud rate clock, an additional break character is queued. If the receiving device is another LPUART, the break characters are received as 0s in all data bits and a framing error (LPUART\_STAT[FE] = 1) occurs.

A break character can also be transmitted by writing to the LPUART\_DATA register with bit 13 set and the data bits clear. This supports transmitting the break character as part of the normal data stream and also allows the DMA to transmit a break character.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for LPUART\_STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the LPUART\_CTRL[TE] bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while LPUART\_CTRL[TE] is cleared, the LPUART transmitter never actually releases control of the LPUART\_TX pin.

An idle character can also be transmitted by writing to the LPUART\_DATA register with bit 13 set and the data bits also set. This supports transmitting the idle character as part of the normal data stream and also allows the DMA to transmit a break character.

The length of the break character is affected by the LPUART\_STAT[BRK13], LPUART\_CTRL[M], LPUART\_BAUD[M10] and LPUART\_BAUD[SNBS] bits as shown below.

**Table 35-1. Break character length**

BRK13	M	M10	SBNS	Break character length
0	0	0	0	10 bit times
0	0	0	1	11 bit times
0	1	0	0	11 bit times
0	1	0	1	12 bit times
0	X	1	0	12 bit times
0	X	1	1	13 bit times
1	0	0	0	13 bit times
1	0	0	1	13 bit times
1	1	0	0	14 bit times
1	1	0	1	14 bit times
1	X	1	0	15 bit times
1	X	1	1	15 bit times

### 35.4.3 Receiver functional description

In this section, the receiver block diagram is a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, different variations of the receiver wakeup function are explained.

The receiver input is inverted by setting LPUART\_STAT[RXINV]. The receiver is enabled by setting the LPUART\_CTRL[RE] bit. Character frames consist of a start bit of logic 0, eight to ten data bits (msb or lsb first), and one or two stop bits of logic 1. For information about 9-bit or 10-bit data mode, refer to [8-bit, 9-bit and 10-bit data modes](#). For the remainder of this discussion, assume the LPUART is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (LPUART\_STAT[RDRF]) status flag is set. If LPUART\_STAT[RDRF] was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the LPUART receiver is double-buffered, the program has one full character time after LPUART\_STAT[RDRF] is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (LPUART\_STAT[RDRF] = 1), it gets the data from the receive data register by reading LPUART\_DATA. Refer to [Interrupts and status flags](#) for details about flag clearing.

### 35.4.3.1 Data sampling technique

The LPUART receiver supports a configurable oversampling rate of between  $4\times$  and  $32\times$  of the baud rate clock for sampling. The receiver starts by taking logic level samples at the oversampling rate times the baud rate to search for a falling edge on the LPUART\_RX serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The oversampling baud rate clock divides the bit time into 4 to 32 segments from 1 to OSR (where OSR is the configured oversampling ratio). When a falling edge is located, three more samples are taken at  $(OSR/2)$ ,  $(OSR/2)+1$ , and  $(OSR/2)+2$  to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a received character. If another falling edge is detected before the receiver is considered synchronized, the receiver restarts the sampling from the first segment.

The receiver then samples each bit time, including the start and stop bits, at  $(OSR/2)$ ,  $(OSR/2)+1$ , and  $(OSR/2)+2$  to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. If any sample in any bit time, including the start and stop bits, in a character frame fails to agree with the logic level for that bit, the noise flag (LPUART\_STAT[NF]) is set when the received character is transferred to the receive data buffer.

When the LPUART receiver is configured to sample on both edges of the baud rate clock, the number of segments in each received bit is effectively doubled (from 1 to  $OSR\times 2$ ). The start and data bits are then sampled at OSR, OSR+1 and OSR+2. Sampling on both edges of the clock must be enabled for oversampling rates of  $4\times$  to  $7\times$  and is optional for higher oversampling rates.

The falling edge detection logic continuously looks for falling edges. If an edge is detected, the sample clock is resynchronized to bit times (unless resynchronization has been disabled). This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

### 35.4.3.2 Receiver wakeup operation

Receiver wakeup and receiver address matching is a hardware mechanism that allows an LPUART receiver to ignore the characters in a message intended for a different receiver.

During receiver wakeup, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up control bit (LPUART\_CTRL[RWU]). When RWU bit and LPUART\_S2[RWUID] bit are set, the status flags associated with the receiver, with the exception of the idle bit, IDLE, are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force LPUART\_CTRL[RWU] to 0 so all receivers wake up in time to look at the first character(s) of the next message.

During receiver address matching, the address matching is performed in hardware and the LPUART receiver will ignore all characters that do not meet the address match requirements.

**Table 35-2. Receiver Wakeup Options**

RWU	MA1   MA2	MATCFG	WAKE:RWUID	Receiver Wakeup
0	0	X	X	Normal operation
1	0	00	00	Receiver wakeup on idle line, IDLE flag not set
1	0	00	01	Receiver wakeup on idle line, IDLE flag set
1	0	00	10	Receiver wakeup on address mark
1	1	11	X0	Receiver wakeup on data match
0	1	00	X0	Address mark address match, IDLE flag not set for discarded characters
0	1	00	X1	Address mark address match, IDLE flag set for discarded characters
0	1	01	X0	Idle line address match
0	1	10	X0	Address match on and address match off, IDLE flag not set for discarded characters
0	1	10	X1	Address match on and address match off, IDLE flag set for discarded characters

### 35.4.3.2.1 Idle-line wakeup

When wake is cleared, the receiver is configured for idle-line wakeup. In this mode, LPUART\_CTRL[RWU] is cleared automatically when the receiver detects a full character time of the idle-line level. The LPUART\_CTRL[M] and LPUART\_BAUD[M10] control bit selects 8-bit to 10-bit data mode and the LPUART\_BAUD[SBNS] bit selects 1-bit or 2-bit stop bit number that determines how many bit times of idle are needed to constitute a full character time, 10 to 13 bit times because of the start and stop bits.

When LPUART\_CTRL[RWU] is one and LPUART\_STAT[RWUID] is zero, the idle condition that wakes up the receiver does not set the LPUART\_STAT[IDLE] flag. The receiver wakes up and waits for the first data character of the next message that sets the LPUART\_STAT[RDRF] flag and generates an interrupt if enabled. When LPUART\_STAT[RWUID] is one, any idle condition sets the LPUART\_STAT[IDLE] flag and generates an interrupt if enabled, regardless of whether LPUART\_CTRL[RWU] is zero or one.

The idle-line type (LPUART\_CTRL[ILT]) control bit selects one of two ways to detect an idle line. When LPUART\_CTRL[ILT] is cleared, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When LPUART\_CTRL[ILT] is set, the idle bit counter does not start until after the stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

### 35.4.3.2.2 Address-mark wakeup

When LPUART\_CTRL[WAKE] is set, the receiver is configured for address-mark wakeup. In this mode, LPUART\_CTRL[RWU] is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character.

Address-mark wakeup allows messages to contain idle characters, but requires the MSB be reserved for use in address frames. The logic 1 in the MSB of an address frame clears the LPUART\_CTRL[RWU] bit before the stop bits are received and sets the LPUART\_STAT[RDRF] flag. In this case, the character with the MSB set is received even though the receiver was sleeping during most of this character time.

### 35.4.3.2.3 Data match wakeup

When LPUART\_CTRL[RWU] is set and LPUART\_BAUD[MATCFG] equals 11, the receiver is configured for data match wakeup. In this mode, LPUART\_CTRL[RWU] is cleared automatically when the receiver detects a character that matches MATCH[MA1] field when BAUD[MAEN1] is set, or that matches MATCH[MA2] when BAUD[MAEN2] is set.

### 35.4.3.2.4 Address Match operation

Address match operation is enabled when the LPUART\_BAUD[MAEN1] or LPUART\_BAUD[MAEN2] bit is set and LPUART\_BAUD[MATCFG] is equal to 00. In this function, a character received by the LPUART\_RX pin with a logic 1 in the bit position immediately preceding the stop bit is considered an address and is compared with the associated MATCH[MA1] or MATCH[MA2] field. The character is only transferred to the receive buffer, and LPUART\_STAT[RDRF] is set, if the comparison matches. All subsequent characters received with a logic 0 in the bit position immediately preceding the stop bit 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 characters with logic zero in the bit position immediately preceding the stop bit are also discarded. If both the LPUART\_BAUD[MAEN1] and LPUART\_BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Address match operation functions in the same way for both MATCH[MA1] and MATCH[MA2] fields.

- If only one of LPUART\_BAUD[MAEN1] and LPUART\_BAUD[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 LPUART\_BAUD[MAEN1] and LPUART\_BAUD[MAEN2] are asserted, a marked address is compared with both match registers and data is transferred only on a match with either register.

### 35.4.3.2.5 Idle Match operation

Idle match operation is enabled when the LPUART\_BAUD[MAEN1] or LPUART\_BAUD[MAEN2] bit is set and LPUART\_BAUD[MATCFG] is equal to 01. In this function, the first character received by the LPUART\_RX pin after an idle line condition is considered an address and is compared with the associated MA1 or MA2 register. The character is only transferred to the receive buffer, and LPUART\_STAT[RDRF] is set, if the comparison matches. All subsequent characters are

considered to be data associated with the address and are transferred to the receive data buffer until the next idle line condition is detected. If no address match occurs then no transfer is made to the receive data buffer, and all following frames until the next idle condition are also discarded. If both the LPUART\_BAUD[MAEN1] and LPUART\_BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

Idle match operation functions in the same way for both MA1 and MA2 registers.

- If only one of LPUART\_BAUD[MAEN1] and LPUART\_BAUD[MAEN2] is asserted, the first character after an idle line is compared only with the associated match register and data is transferred to the receive data buffer only on a match.
- If LPUART\_BAUD[MAEN1] and LPUART\_BAUD[MAEN2] are asserted, the first character after an idle line is compared with both match registers and data is transferred only on a match with either register.

#### 35.4.3.2.6 Match On Match Off operation

Match on, match off operation is enabled when both LPUART\_BAUD[MAEN1] and LPUART\_BAUD[MAEN2] are set and LPUART\_BAUD[MATCFG] is equal to 10. In this function, a character received by the LPUART\_RX pin that matches MATCH[MA1] is received and transferred to the receive buffer, and LPUART\_STAT[RDRF] is set. All subsequent characters are considered to be data and are also transferred to the receive data buffer, until a character is received that matches MATCH[MA2] register. The character that matches MATCH[MA2] and all following characters are discarded, this continues until another character that matches MATCH[MA1] is received. If both the LPUART\_BAUD[MAEN1] and LPUART\_BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive data buffer.

#### NOTE

Match on, match off operation requires both LPUART\_BAUD[MAEN1] and LPUART\_BAUD[MAEN2] to be asserted.

### 35.4.4 Additional LPUART functions

The following sections describe additional LPUART functions.

### 35.4.4.1 8-bit, 9-bit and 10-bit data modes

The LPUART transmitter and receiver can be configured to operate in 9-bit data mode by setting the LPUART\_CTRL[M] or 10-bit data mode by setting LPUART\_CTRL[M10]. In 9-bit mode, there is a ninth data bit in 10-bit mode there is a tenth data bit. For the transmit data buffer, these bits are stored in LPUART\_CTRL[T8] and LPUART\_CTRL[T9]. For the receiver, these bits are held in LPUART\_CTRL[R8] and LPUART\_CTRL[R9]. They are also accessible via 16-bit or 32-bit accesses to the LPUART\_DATA register.

For coherent 8-bit writes to the transmit data buffer, write to LPUART\_CTRL[T8] and LPUART\_CTRL[T9] before writing to LPUART\_DATA[7:0]. For 16-bit and 32-bit writes to the LPUART\_DATA register all 10 transmit bits are written to the transmit data buffer at the same time.

If the bit values to be transmitted as the ninth and tenth bit of a new character are the same as for the previous character, it is not necessary to write to LPUART\_CTRL[T8] and LPUART\_CTRL[T9] again. When data is transferred from the transmit data buffer to the transmit shifter, the value in LPUART\_CTRL[T8] and LPUART\_CTRL[T9] is copied at the same time data is transferred from LPUART\_DATA[7:0] to the shifter.

The 9-bit data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. The 10-bit data mode is typically used with parity and address-mark wakeup so the ninth data bit can serve as the wakeup bit and the tenth bit as the parity bit. In custom protocols, the ninth and/or tenth bits can also serve as software-controlled markers.

### 35.4.4.2 Idle length

An idle character is a character where the start bit, all data bits and stop bits are in the mark position. The CTRL[ILT] register can be configured to start detecting an idle character from the previous start bit (any data bits and stop bits count towards the idle character detection) or from the previous stop bit.

The number of idle characters that must be received before an idle line condition is detected can also be configured using the CTRL[IDLECFG] field. This field configures the number of idle characters that must be received before the STAT[IDLE] flag is set, the STAT[RAF] flag is cleared and the DATA[IDLINE] flag is set with the next received character.



Idle-line wakeup and idle match operation are also affected by the CTRL[IDLECFG] field. When address match or match on/off operation is enabled, setting the STAT[RWUID] bit will cause any discarded characters to be treated as if they were idle characters.

### 35.4.4.3 Loop mode

When LPUART\_CTRL[LOOPS] is set, the LPUART\_CTRL[RSRC] bit in the same register chooses between loop mode (LPUART\_CTRL[RSRC] = 0) or single-wire mode (LPUART\_CTRL[RSRC] = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the LPUART\_RX pin is not used by the LPUART.

### 35.4.4.4 Single-wire operation

When LPUART\_CTRL[LOOPS] is set, the RSRC bit in the same register chooses between loop mode (LPUART\_CTRL[RSRC] = 0) or single-wire mode (LPUART\_CTRL[RSRC] = 1). Single-wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the LPUART\_TX pin (the LPUART\_RX pin is not used).

In single-wire mode, the LPUART\_CTRL[TXDIR] bit controls the direction of serial data on the LPUART\_TX pin. When LPUART\_CTRL[TXDIR] is cleared, the LPUART\_TX pin is an input to the receiver and the transmitter is temporarily disconnected from the LPUART\_TX pin so an external device can send serial data to the receiver. When LPUART\_CTRL[TXDIR] is set, the LPUART\_TX pin is an output driven by the transmitter, the internal loop back connection is disabled, and as a result the receiver cannot receive characters that are sent out by the transmitter.

### 35.4.5 Interrupts and status flags

The LPUART transmitter has two status flags that can optionally generate hardware interrupt requests. Transmit data register empty LPUART\_STAT[TDRE]) indicates when there is room in the transmit data buffer to write another transmit character to LPUART\_DATA. If the transmit interrupt enable LPUART\_CTRL[TIE]) bit is set, a hardware interrupt is requested when LPUART\_STAT[TDRE] is set. Transmit complete (LPUART\_STAT[TC]) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with LPUART\_TX at the inactive level. This

## Functional description

flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (LPUART\_CTRL[TCIE]) bit is set, a hardware interrupt is requested when LPUART\_STAT[TC] is set. Instead of hardware interrupts, software polling may be used to monitor the LPUART\_STAT[TDRE] and LPUART\_STAT[TC] status flags if the corresponding LPUART\_CTRL[TIE] or LPUART\_CTRL[TCIE] local interrupt masks are cleared.

When a program detects that the receive data register is full (LPUART\_STAT[RDRF] = 1), it gets the data from the receive data register by reading LPUART\_DATA. The LPUART\_STAT[RDRF] flag is cleared by reading LPUART\_DATA.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the LPUART\_RX line remains idle for an extended period of time. IDLE is cleared by writing 1 to the LPUART\_STAT[IDLE] flag. After LPUART\_STAT[IDLE] has been cleared, it cannot become set again until the receiver has received at least one new character and has set LPUART\_STAT[RDRF].

If the associated error was detected in the received character that caused LPUART\_STAT[RDRF] to be set, the error flags - noise flag (LPUART\_STAT[NF]), framing error (LPUART\_STAT[FE]), and parity error flag (LPUART\_STAT[PF]) - are set at the same time as LPUART\_STAT[RDRF]. These flags are not set in overrun cases.

If LPUART\_STAT[RDRF] was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (LPUART\_STAT[OR]) flag is set instead of the data along with any associated NF, FE, or PF condition is lost.

If the received character matches the contents of MATCH[MA1] and/or MATCH[MA2] then the LPUART\_STAT[MA1F] and/or LPUART\_STAT[MA2F] flags are set at the same time that LPUART\_STAT[RDRF] is set.

At any time, an active edge on the LPUART\_RX serial data input pin causes the LPUART\_STAT[RXEDGIF] flag to set. The LPUART\_STAT[RXEDGIF] flag is cleared by writing a 1 to it. This function depends on the receiver being enabled (LPUART\_CTRL[RE] = 1).

# Chapter 36

## Universal Asynchronous Receiver/ Transmitter(UART)

### 36.1 Chip-specific UART information

#### 36.1.1 UART2 Overview

This device contains a basic universal asynchronous receiver/transmitter (UART) modules with DMA function support. Generally, these modules are used in RS-232, RS-485, and other communications. This module supports LIN Slave operation.

### 36.2 Introduction

The UART allows asynchronous serial communication with peripheral devices and CPUs.

#### 36.2.1 Features

The UART includes the following features:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- 13-bit baud rate selection with /32 fractional divide, based on the module clock frequency
- Programmable 8-bit or 9-bit data format
- Separately enabled transmitter and receiver

## Introduction

- 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
- Support for ISO 7816 protocol to interface with SIM cards and smart cards
  - Support for T=0 and T=1 protocols
  - Automatic retransmission of NACK'd packets with programmable retry threshold
  - Support for 11 and 12 ETU transfers
  - Detection of initial packet and automated transfer parameter programming
  - Interrupt-driven operation with seven ISO-7816 specific interrupts:
    - Wait time violated
    - Character wait time violated
    - Block wait time violated
    - Initial frame detected
    - Transmit error threshold exceeded
    - Receive error threshold exceeded
    - Guard time violated
- Interrupt-driven operation with flags, not specific to ISO-7816 support
  - 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
- Receiver framing error detection
- Hardware parity generation and checking
- 1/16 bit-time noise detection
- DMA interface

## 36.2.2 Modes of operation

The UART functions in the same way in all the normal modes.

It has the following low power mode:

- Stop mode

### 36.2.2.1 Run mode

This is the normal mode of operation.

### 36.2.2.2 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. Entering or leaving Stop mode does not initiate any power down or power up procedures for the ISO-7816 smartcard interface.

### 36.3 UART signal descriptions

The UART signals are shown in the following table.

**Table 36-1. UART signal descriptions**

Signal	Description	I/O
RXD	Receive data	I
TXD	Transmit data	O

#### 36.3.1 Detailed signal descriptions

The detailed signal descriptions of the UART are shown in the following table.

**Table 36-2. UART—Detailed signal descriptions**

Signal	I/O	Description	
RXD	I	Receive data. Serial data input to receiver.	
		<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.
TXD	O	Transmit data. Serial data output from transmitter.	
		<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.

### 36.4 Memory map and registers

This section provides a detailed description of all memory and registers.

Only byte accesses are supported.

#### UART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4006_C000	UART Baud Rate Registers: High (UART2_BDH)	8	R/W	00h	<a href="#">36.4.1/632</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_C001	UART Baud Rate Registers: Low (UART2_BDL)	8	R/W	04h	<a href="#">36.4.2/633</a>
4006_C002	UART Control Register 1 (UART2_C1)	8	R/W	00h	<a href="#">36.4.3/633</a>
4006_C003	UART Control Register 2 (UART2_C2)	8	R/W	00h	<a href="#">36.4.4/635</a>
4006_C004	UART Status Register 1 (UART2_S1)	8	R	C0h	<a href="#">36.4.5/637</a>
4006_C005	UART Status Register 2 (UART2_S2)	8	R/W	00h	<a href="#">36.4.6/639</a>
4006_C006	UART Control Register 3 (UART2_C3)	8	R/W	00h	<a href="#">36.4.7/641</a>
4006_C007	UART Data Register (UART2_D)	8	R/W	00h	<a href="#">36.4.8/642</a>
4006_C008	UART Match Address Registers 1 (UART2_MA1)	8	R/W	00h	<a href="#">36.4.9/643</a>
4006_C009	UART Match Address Registers 2 (UART2_MA2)	8	R/W	00h	<a href="#">36.4.10/644</a>
4006_C00A	UART Control Register 4 (UART2_C4)	8	R/W	00h	<a href="#">36.4.11/644</a>
4006_C00B	UART Control Register 5 (UART2_C5)	8	R/W	00h	<a href="#">36.4.12/645</a>
4006_C018	UART 7816 Control Register (UART2_C7816)	8	R/W	00h	<a href="#">36.4.13/646</a>
4006_C019	UART 7816 Interrupt Enable Register (UART2_IE7816)	8	R/W	00h	<a href="#">36.4.14/647</a>
4006_C01A	UART 7816 Interrupt Status Register (UART2_IS7816)	8	R/W	00h	<a href="#">36.4.15/649</a>
4006_C01B	UART 7816 Wait Parameter Register (UART2_WP7816)	8	R/W	00h	<a href="#">36.4.16/651</a>
4006_C01C	UART 7816 Wait N Register (UART2_WN7816)	8	R/W	00h	<a href="#">36.4.17/651</a>
4006_C01D	UART 7816 Wait FD Register (UART2_WF7816)	8	R/W	01h	<a href="#">36.4.18/652</a>
4006_C01E	UART 7816 Error Threshold Register (UART2_ET7816)	8	R/W	00h	<a href="#">36.4.19/652</a>
4006_C01F	UART 7816 Transmit Length Register (UART2_TL7816)	8	R/W	00h	<a href="#">36.4.20/653</a>
4006_C03A	UART 7816 ATR Duration Timer Register A (UART2_AP7816A_T0)	8	R/W	00h	<a href="#">36.4.21/653</a>
4006_C03B	UART 7816 ATR Duration Timer Register B (UART2_AP7816B_T0)	8	R/W	00h	<a href="#">36.4.22/654</a>
4006_C03C	UART 7816 Wait Parameter Register A (UART2_WP7816A_T0)	8	R/W	00h	<a href="#">36.4.23/655</a>
4006_C03C	UART 7816 Wait Parameter Register A (UART2_WP7816A_T1)	8	R/W	00h	<a href="#">36.4.24/655</a>
4006_C03D	UART 7816 Wait Parameter Register B (UART2_WP7816B_T0)	8	R/W	14h	<a href="#">36.4.25/656</a>
4006_C03D	UART 7816 Wait Parameter Register B (UART2_WP7816B_T1)	8	R/W	14h	<a href="#">36.4.26/656</a>

Table continues on the next page...

UART memory map (continued)

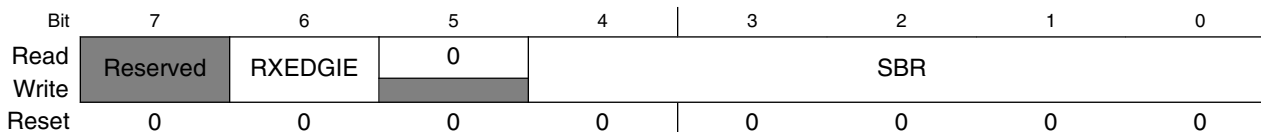
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4006_C03E	UART 7816 Wait and Guard Parameter Register (UART2_WGP7816_T1)	8	R/W	06h	36.4.27/657
4006_C03F	UART 7816 Wait Parameter Register C (UART2_WP7816C_T1)	8	R/W	0Bh	36.4.28/657

36.4.1 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: 4006\_C000h base + 0h offset = 4006\_C000h



UARTx\_BDH field descriptions

Field	Description
7 Reserved	Reserved. This field is reserved.
6 RXEDGIE	RxD Input Active Edge Interrupt Enable Enables the receive input active edge, RXEDGIF, to generate interrupt requests. 0 Hardware interrupts from RXEDGIF disabled using polling. 1 RXEDGIF interrupt request enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SBR	UART Baud Rate Bits The baud rate for the UART is determined by the 13 SBR fields. See <a href="#">Baud rate generation</a> for details. <b>NOTE:</b> <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>



### 36.4.2 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: 4006\_C000h base + 1h offset = 4006\_C001h

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

#### UARTx\_BDL field descriptions

Field	Description
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>

### 36.4.3 UART Control Register 1 (UARTx\_C1)

This read/write register controls various optional features of the UART system.

Address: 4006\_C000h base + 2h offset = 4006\_C002h

Bit	7	6	5	4	3	2	1	0
Read	LOOPS	Reserved	RSRC	M	WAKE	ILT	PE	PT
Write	LOOPS	Reserved	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

#### 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 Reserved	Reserved. This field is reserved.
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 This field must be set when C7816[ISO_7816E] is set/enabled. 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. This field must be set when C7816[ISO_7816E] is set/enabled. 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. This field must be cleared when C7816[ISO_7816E] is set/enabled.
0	Even parity.
1	Odd parity.

**36.4.4 UART Control Register 2 (UARTx\_C2)**

This register can be read or written at any time.

Address: 4006\_C000h base + 3h offset = 4006\_C003h

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	Transmitter Interrupt or DMA Transfer Enable.  Enables S1[TDRE] to generate interrupt requests or DMA transfer requests, based on the state of C5[TDMAS]. <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.  0 TDRE interrupt and DMA transfer requests disabled. 1 TDRE interrupt or DMA transfer requests enabled.
6 TCIE	Transmission Complete Interrupt Enable  Enables the transmission complete flag, S1[TC], to generate interrupt requests .  0 TC interrupt requests disabled. 1 TC interrupt requests enabled.
5 RIE	Receiver Full Interrupt or DMA Transfer Enable  Enables S1[RDRF] to generate interrupt requests or DMA transfer requests, based on the state of C5[RDMAS].  0 RDRF interrupt and DMA transfer requests disabled. 1 RDRF interrupt or DMA transfer requests enabled.
4 ILIE	Idle Line Interrupt Enable  Enables the idle line flag, S1[IDLE], to generate interrupt requests

*Table continues on the next page...*

## UARTx\_C2 field descriptions (continued)

Field	Description
	0 IDLE interrupt requests disabled. 1 IDLE interrupt requests enabled.
3 TE	Transmitter Enable  Enables the UART transmitter. TE can be used to queue an idle preamble by clearing and then setting TE. When C7816[ISO_7816E] is set/enabled and C7816[TTYTYPE] = 1, this field is automatically cleared after the requested block has been transmitted. This condition is detected when TL7816[TLEN] = 0 and four additional characters are transmitted.  0 Transmitter off. 1 Transmitter on.
2 RE	Receiver Enable  Enables the UART receiver.  0 Receiver off. 1 Receiver on.
1 RWU	Receiver Wakeup Control  This field can be set to place the UART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when C1[WAKE] is clear or an address match when C1[WAKE] is set. This field must be cleared when C7816[ISO_7816E] is set.  <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 after an IDLE is detected before IDLE is allowed to reasserted.  0 Normal operation. 1 RWU enables the wakeup function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU.
0 SBK	Send Break  Toggling SBK sends one break character from the following: See <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. <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> This field must be cleared when C7816[ISO_7816E] is set.  0 Normal transmitter operation. 1 Queue break characters to be sent.

### 36.4.5 UART Status Register 1 (UARTx\_S1)

The S1 register provides inputs to the MCU for generation of UART interrupts or DMA requests. This register can also be polled by the MCU to check the status of its fields. To clear a flag, the status register should be read followed by a read or write to D register, depending on the interrupt flag type. Other instructions can be executed between the two steps as long the handling of I/O is not compromised, but the order of operations is important for flag clearing. When a flag is configured to trigger a DMA request, assertion of the associated DMA done signal from the DMA controller clears the flag.

#### NOTE

- If the condition that results in the assertion of the flag, interrupt, or DMA request is not resolved prior to clearing the flag, the flag, and interrupt/DMA request, reasserts.

Address: 4006\_C000h base + 4h offset = 4006\_C004h

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 transmit data register (D) is empty. To clear TDRE, read S1 when TDRE is set and then write to the UART data register (D).</p> <p>0 Transmit data buffer is full. 1 Transmit data buffer is empty.</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: When C7816[ISO_7816E] is set/enabled, this field is set after any NACK signal has been received, but prior to any corresponding guard times expiring.</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>
5 RDRF	<p>Receive Data Register Full Flag</p> <p>RDRF is set when the receive buffer (D) is full. To clear RDRF, read S1 when RDRF is set and then read D.</p>

Table continues on the next page...

## UARTx\_S1 field descriptions (continued)

Field	Description
	0 Receive data buffer is empty. 1 Receive data buffer is full.
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). 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>Idle detection is not supported when 7816E is set/enabled and hence this flag is ignored.</p> <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. In 7816 mode, it is possible to configure a NACK to be returned by programming C7816[ONACK].</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. To clear NF, read S1 and then read D.</p> <p>0 No noise detected.            1 Noise detected in the received character in D.</p>
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. 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. Framing errors are not supported when 7816E is set/enabled. However, if this flag is set, data is still not received in 7816 mode.</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. To clear PF, read S1 and then read D.</p>

Table continues on the next page...

## UARTx\_S1 field descriptions (continued)

Field	Description
0	No parity error detected.
1	Parity error.

## 36.4.6 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: 4006\_C000h base + 5h offset = 4006\_C005h

Bit	7	6	5	4	3	2	1	0
Read	0	RXEDGIF	MSBF	RXINV	RWUID	BRK13	Reserved	RAF
Write		w1c						
Reset	0	0	0	0	0	0	0	0

## UARTx\_S2 field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag  RXEDGIF is set when an active edge occurs on the RxD pin. The active edge is falling if RXINV = 0, and rising if RXINV=1. RXEDGIF is cleared by writing a 1 to it. See for additional details. <a href="#">RXEDGIF description</a>  <b>NOTE:</b> The active edge is detected only in two wire mode and on receiving data coming from the RxD pin.  0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
5 MSBF	Most Significant Bit First  Setting this field reverses the order of the bits that are transmitted and received on the wire. This field does not affect the polarity of the bits, the location of the parity bit, or the location of the start or stop bits. This field is automatically set when C7816[INIT] and C7816[ISO7816E] are enabled and an initial character is detected in T = 0 protocol mode.  0 LSB (bit0) is the first bit that is transmitted following the start bit. Further, the first bit received after the start bit is identified as bit0. 1 MSB (bit8, bit7 or bit6) is the first bit that is transmitted following the start bit, depending on the setting of C1[M] and C1[PE]. Further, the first bit received after the start bit is identified as bit8, bit7, or bit6, depending on the setting of C1[M] and C1[PE].
4 RXINV	Receive Data Inversion

Table continues on the next page...

## UARTx\_S2 field descriptions (continued)

Field	Description
	<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. This field is automatically set when C7816[INIT] and C7816[ISO7816E] are enabled and an initial character is detected in T = 0 protocol mode.</p> <p><b>NOTE:</b> Setting RXINV inverts the RxD input for data bits, start and stop bits, break, and idle. When C7816[ISO7816E] is set/enabled, only the data bits and the parity bit are inverted.</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]. This field must be cleared when C7816[ISO7816E] is set/enabled.</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> <p>0 Break character is 10, 11, or 12 bits long. 1 Break character is 13 or 14 bits long.</p>
1 Reserved	<p>Reserved.</p> <p>This field is reserved.</p>
0 RAF	<p>Receiver Active Flag</p> <p>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 when C7816[ISO7816E] is cleared/disabled. When C7816[ISO7816E] is enabled, the RAF is cleared if the C7816[TTYTYPE] = 0 expires or the C7816[TTYTYPE] = 1 expires.</p> <p><b>NOTE:</b> In case C7816[ISO7816E] is set and C7816[TTYTYPE] = 0, it is possible to configure the guard time to 12. However, if a NACK is required to be transmitted, the data transfer actually takes 13 ETU with the 13th ETU slot being a inactive buffer. Therefore, in this situation, the RAF may deassert one ETU prior to actually being inactive.</p> <p>0 UART receiver idle/inactive waiting for a start bit. 1 UART receiver active, RxD input not idle.</p>



### 36.4.7 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: 4006\_C000h base + 6h offset = 4006\_C006h

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	<p>Received Bit 8</p> <p>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.</p>
6 T8	<p>Transmit Bit 8</p> <p>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.</p> <p><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.</p> <p>To correctly transmit the 9th bit, write UARTx_C3[T8] to the desired value, then write the UARTx_D register with the remaining data.</p>
5 TXDIR	<p>Transmitter Pin Data Direction in Single-Wire mode</p> <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. When C7816[ISO7816E] is set/enabled and C7816[TTYTYPE] = 1, this field is automatically cleared after the requested block is transmitted. This condition is detected when TL7816[TLEN] = 0 and 4 additional characters are transmitted. Additionally, if C7816[ISO7816E] is set/enabled and C7816[TTYTYPE] = 0 and a NACK is being transmitted, the hardware automatically overrides this field as needed. In this situation, TXDIR does not reflect the temporary state associated with the NACK.</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. This field is automatically set when C7816[INIT] and C7816[ISO7816E] are enabled and an initial character is detected in T = 0 protocol mode.</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. When C7816[ISO7816E] is set/enabled then only the transmitted data bits and parity bit are inverted.</p>

Table continues on the next page...

**UARTx\_C3 field descriptions (continued)**

Field	Description
	0 Transmit data is not inverted. 1 Transmit data is inverted.
3 ORIE	Overrun Error Interrupt Enable  Enables the overrun error flag, S1[OR], to generate interrupt requests.  0 OR interrupts are disabled. 1 OR interrupt requests are enabled.
2 NEIE	Noise Error Interrupt Enable  Enables the noise flag, S1[NF], to generate interrupt requests.  0 NF interrupt requests are disabled. 1 NF interrupt requests are enabled.
1 FEIE	Framing Error Interrupt Enable  Enables the framing error flag, S1[FE], to generate interrupt requests.  0 FE interrupt requests are disabled. 1 FE interrupt requests are enabled.
0 PEIE	Parity Error Interrupt Enable  Enables the parity error flag, S1[PF], to generate interrupt requests.  0 PF interrupt requests are disabled. 1 PF interrupt requests are enabled.

**36.4.8 UART Data Register (UARTx\_D)**

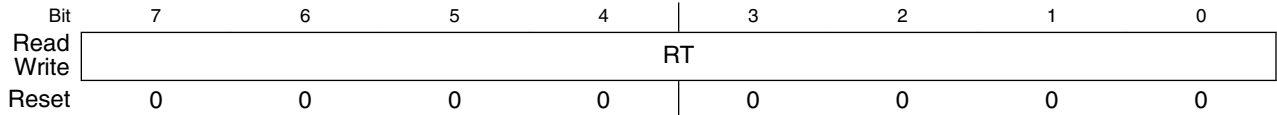
This register is actually two separate registers. Reads return the contents of the read-only receive data register and writes go to the write-only transmit data register.

**NOTE**

- In 8-bit or 9-bit data format, only UART data register (D) needs to be accessed to clear the S1[RDRF] bit . The C3 register needs to be read, prior to the D register, only if the ninth bit of data 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: 4006\_C000h base + 7h offset = 4006\_C007h



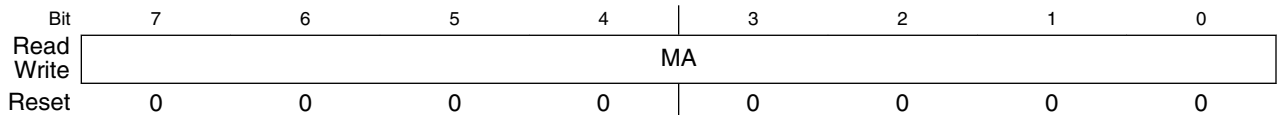
**UARTx\_D field descriptions**

Field	Description
RT	Reads return the contents of the read-only receive data register and writes go to the write-only transmit data register.

**36.4.9 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: 4006\_C000h base + 8h offset = 4006\_C008h



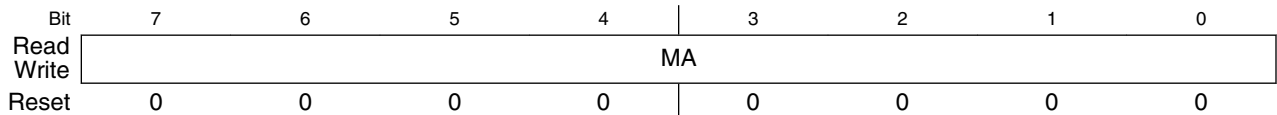
**UARTx\_MA1 field descriptions**

Field	Description
MA	Match Address

### 36.4.10 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: 4006\_C000h base + 9h offset = 4006\_C009h

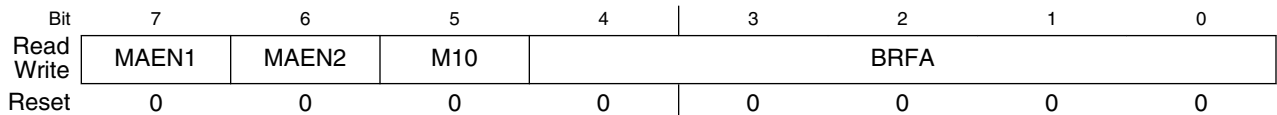


**UARTx\_MA2 field descriptions**

Field	Description
MA	Match Address

### 36.4.11 UART Control Register 4 (UARTx\_C4)

Address: 4006\_C000h base + Ah offset = 4006\_C00Ah



**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.</p> <p>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. This field must be cleared when C7816[ISO7816E] is set/enabled.</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.</p> <p>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. This field must be cleared when C7816[ISO7816E] is set/enabled.</p>

*Table continues on the next page...*

## UARTx\_C4 field descriptions (continued)

Field	Description
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. If M10 is set, then both C1[M] and C1[PE] must also be set. This field must be cleared when C7816[ISO7816E] is set/enabled.</p> <p>See <a href="#">Data format (non ISO-7816)</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>
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>

## 36.4.12 UART Control Register 5 (UARTx\_C5)

Address: 4006\_C000h base + Bh offset = 4006\_C00Bh

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> <p>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.</p>
6 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
5 RDMAS	<p>Receiver Full DMA Select</p> <p>Configures the receiver data register full flag, S1[RDRF], to generate interrupt or DMA requests if C2[RIE] is set.</p> <p><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.</p>

*Table continues on the next page...*

### UARTx\_C5 field descriptions (continued)

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

### 36.4.13 UART 7816 Control Register (UARTx\_C7816)

The C7816 register is the primary control register for ISO-7816 specific functionality. This register is specific to 7816 functionality and the values in this register have no effect on UART operation and should be ignored if ISO\_7816E is not set/enabled. This register may be read at any time but values must be changed only when ISO\_7816E is not set.

Address: 4006\_C000h base + 18h offset = 4006\_C018h

Bit	7	6	5	4	3	2	1	0
Read	0			ONACK	ANACK	INIT	TTYTYPE	ISO_7816E
Write	0			0	0	0	0	0
Reset	0	0	0	0	0	0	0	0

### UARTx\_C7816 field descriptions

Field	Description
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 ONACK	Generate NACK on Overflow  When this field is set, the receiver automatically generates a NACK response if a receive buffer overrun occurs, as indicated by S1[OR]. In many systems, this results in the transmitter resending the packet that overflowed until the retransmit threshold for that transmitter is reached. A NACK is generated only if TTYTYPE=0. This field operates independently of ANACK. See . <a href="#">Overrun NACK considerations</a>  0 The received data does not generate a NACK when the receipt of the data results in an overflow event. 1 If the receiver buffer overflows, a NACK is automatically sent on a received character.
3 ANACK	Generate NACK on Error  When this field is set, the receiver automatically generates a NACK response if a parity error occurs or if INIT is set and an invalid initial character is detected. A NACK is generated only if TTYTYPE = 0. If ANACK is set, the UART attempts to retransmit the data indefinitely. To stop retransmission attempts, clear C2[TE] or ISO_7816E and do not set until S1[TC] sets C2[TE] again.

Table continues on the next page...

## UARTx\_C7816 field descriptions (continued)

Field	Description
	0 No NACK is automatically generated. 1 A NACK is automatically generated if a parity error is detected or if an invalid initial character is detected.
2 INIT	Detect Initial Character  When this field is set, all received characters are searched for a valid initial character. If an invalid initial character is identified, and ANACK is set, a NACK is sent. All received data is discarded and error flags blocked (S1[NF], S1[OR], S1[FE], S1[PF], IS7816[WT], IS7816[CWT], IS7816[BWT], IS7816[ADT], IS7816[GTV]) until a valid initial character is detected. Upon detecting a valid initial character, the configuration values S2[MSBF], C3[TXINV], and S2[RXINV] are automatically updated to reflect the initial character that was received. The actual INIT data value is not stored in the receive buffer. Additionally, upon detection of a valid initial character, IS7816[INITD] is set and an interrupt issued as programmed by IE7816[INITDE]. When a valid initial character is detected, INIT is automatically cleared. This Initial Character Detect feature is supported only in T = 0 protocol mode.  0 Normal operating mode. Receiver does not seek to identify initial character. 1 Receiver searches for initial character.
1 TTYE	Transfer Type  Indicates the transfer protocol being used.  See <a href="#">ISO-7816 / smartcard support</a> for more details.  0 T = 0 per the ISO-7816 specification. 1 T = 1 per the ISO-7816 specification.
0 ISO_7816E	ISO-7816 Functionality Enabled  Indicates that the UART is operating according to the ISO-7816 protocol.  <b>NOTE:</b> This field must be modified only when no transmit or receive is occurring. If this field is changed during a data transfer, the data being transmitted or received may be transferred incorrectly.  0 ISO-7816 functionality is turned off/not enabled. 1 ISO-7816 functionality is turned on/enabled.

### 36.4.14 UART 7816 Interrupt Enable Register (UARTx\_IE7816)

The IE7816 register controls which flags result in an interrupt being issued. This register is specific to 7816 functionality, the corresponding flags that drive the interrupts are not asserted when 7816E is not set/enabled. However, these flags may remain set if they are asserted while 7816E was set and not subsequently cleared. This register may be read or written to at any time.

Address: 4006\_C000h base + 19h offset = 4006\_C019h

Bit	7	6	5	4	3	2	1	0
Read	WTE	CWTE	BWTE	INITDE	ADTE	GTVE	TXTE	RXTE
Write								
Reset	0	0	0	0	0	0	0	0

## UARTx\_IE7816 field descriptions

Field	Description
7 WTE	Wait Timer Interrupt Enable 0 The assertion of IS7816[WT] does not result in the generation of an interrupt. 1 The assertion of IS7816[WT] results in the generation of an interrupt.
6 CWTE	Character Wait Timer Interrupt Enable 0 The assertion of IS7816[CWT] does not result in the generation of an interrupt. 1 The assertion of IS7816[CWT] results in the generation of an interrupt.
5 BWTE	Block Wait Timer Interrupt Enable 0 The assertion of IS7816[BWT] does not result in the generation of an interrupt. 1 The assertion of IS7816[BWT] results in the generation of an interrupt.
4 INITDE	Initial Character Detected Interrupt Enable 0 The assertion of IS7816[INITD] does not result in the generation of an interrupt. 1 The assertion of IS7816[INITD] results in the generation of an interrupt.
3 ADTE	ATR Duration Timer Interrupt Enable 0 The assertion of IS7816[ADT] does not result in the generation of an interrupt. 1 The assertion of IS7816[ADT] results in the generation of an interrupt.
2 GTVE	Guard Timer Violated Interrupt Enable 0 The assertion of IS7816[GTV] does not result in the generation of an interrupt. 1 The assertion of IS7816[GTV] results in the generation of an interrupt.
1 TXTE	Transmit Threshold Exceeded Interrupt Enable 0 The assertion of IS7816[TXT] does not result in the generation of an interrupt. 1 The assertion of IS7816[TXT] results in the generation of an interrupt.
0 RXTE	Receive Threshold Exceeded Interrupt Enable 0 The assertion of IS7816[RXT] does not result in the generation of an interrupt. 1 The assertion of IS7816[RXT] results in the generation of an interrupt.



### 36.4.15 UART 7816 Interrupt Status Register (UARTx\_IS7816)

The IS7816 register provides a mechanism to read and clear the interrupt flags. All flags/interrupts are cleared by writing a 1 to the field location. Writing a 0 has no effect. All bits are "sticky", meaning they indicate that only the flag condition that occurred since the last time the bit was cleared, not that the condition currently exists. The status flags are set regardless of whether the corresponding field in the IE7816 is set or cleared. The IE7816 controls only if an interrupt is issued to the host processor. This register is specific to 7816 functionality and the values in this register have no affect on UART operation and should be ignored if 7816E is not set/enabled. This register may be read or written at anytime.

Address: 4006\_C000h base + 1Ah offset = 4006\_C01Ah

Bit	7	6	5	4	3	2	1	0
Read	WT	CWT	BWT	INITD	ADT	GTV	TXT	RXT
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

#### UARTx\_IS7816 field descriptions

Field	Description
7 WT	<p>Wait Timer Interrupt</p> <p>Indicates that the wait time, the time between the leading edge of a character being transmitted and the leading edge of the next response character, has exceeded the programmed value. This flag asserts only when C7816[TTYPE] = 0. This interrupt is cleared by writing 1.</p> <p>0 Wait time (WT) has not been violated. 1 Wait time (WT) has been violated.</p>
6 CWT	<p>Character Wait Timer Interrupt</p> <p>Indicates that the character wait time, the time between the leading edges of two consecutive characters in a block, has exceeded the programmed value. This flag asserts only when C7816[TTYPE] = 1. This interrupt is cleared by writing 1.</p> <p>0 Character wait time (CWT) has not been violated. 1 Character wait time (CWT) has been violated.</p>
5 BWT	<p>Block Wait Timer Interrupt</p> <p>Indicates that the block wait time, the time between the leading edge of first received character of a block and the leading edge of the last character of the previously transmitted block, has exceeded the programmed value. This flag asserts only when C7816[TTYPE] = 1. This interrupt is cleared by writing 1.</p> <p>0 Block wait time (BWT) has not been violated. 1 Block wait time (BWT) has been violated.</p>
4 INITD	<p>Initial Character Detected Interrupt</p> <p>Indicates that a valid initial character is received. This interrupt is cleared by writing 1.</p>

*Table continues on the next page...*

## UARTx\_IS7816 field descriptions (continued)

Field	Description
	0 A valid initial character has not been received. 1 A valid initial character has been received.
3 ADT	<b>ATR Duration Time Interrupt</b>  Indicates that the ATR duration time, the time between the leading edge of the TS character being received and the leading edge of the next response character, has exceeded the programmed value. This flag asserts only when C7816[TTYTYPE] = 0. This interrupt is cleared by writing 1.  0 ATR Duration time (ADT) has not been violated. 1 ATR Duration time (ADT) has been violated.
2 GTV	<b>Guard Timer Violated Interrupt</b>  Indicates that one or more of the character guard time, block guard time, or guard time are violated. This interrupt is cleared by writing 1.  0 A guard time (GT, CGT, or BGT) has not been violated. 1 A guard time (GT, CGT, or BGT) has been violated.
1 TXT	<b>Transmit Threshold Exceeded Interrupt</b>  Indicates that the transmit NACK threshold has been exceeded as indicated by ET7816[TXTHRESHOLD]. Regardless of whether this flag is set, the UART continues to retransmit indefinitely. This flag asserts only when C7816[TTYTYPE] = 0. If 7816E is cleared/disabled, ANACK is cleared/disabled, C2[TE] is cleared/disabled, C7816[TTYTYPE] = 1, or packet is transferred without receiving a NACK, the internal NACK detection counter is cleared and the count restarts from zero on the next received NACK. This interrupt is cleared by writing 1.  0 The number of retries and corresponding NACKS does not exceed the value in ET7816[TXTHRESHOLD]. 1 The number of retries and corresponding NACKS exceeds the value in ET7816[TXTHRESHOLD].
0 RXT	<b>Receive Threshold Exceeded Interrupt</b>  Indicates that there are more than ET7816[RXTHRESHOLD] consecutive NACKS generated in response to parity errors on received data. This flag requires ANACK to be set. Additionally, this flag asserts only when C7816[TTYTYPE] = 0. Clearing this field also resets the counter keeping track of consecutive NACKS. The UART will continue to attempt to receive data regardless of whether this flag is set. If 7816E is cleared/disabled, RE is cleared/disabled, C7816[TTYTYPE] = 1, or packet is received without needing to issue a NACK, the internal NACK detection counter is cleared and the count restarts from zero on the next transmitted NACK. This interrupt is cleared by writing 1.  0 The number of consecutive NACKS generated as a result of parity errors and buffer overruns is less than or equal to the value in ET7816[RXTHRESHOLD]. 1 The number of consecutive NACKS generated as a result of parity errors and buffer overruns is greater than the value in ET7816[RXTHRESHOLD].

### 36.4.16 UART 7816 Wait Parameter Register (UARTx\_WP7816)

The WP7816 register contains the WTX variable used in the generation of the block wait timer. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 1Bh offset = 4006\_C01Bh

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

#### UARTx\_WP7816 field descriptions

Field	Description
WTX	Wait Time Multiplier (C7816[TTYE] = 1)  Used to calculate the value used for the BWT counter. It represents a value between 0 and 255. This value is used only when C7816[TTYE] = 1. See <a href="#">Wait time and guard time parameters</a> .

### 36.4.17 UART 7816 Wait N Register (UARTx\_WN7816)

The WN7816 register contains a parameter that is used in the calculation of the guard time counter. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 1Ch offset = 4006\_C01Ch

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

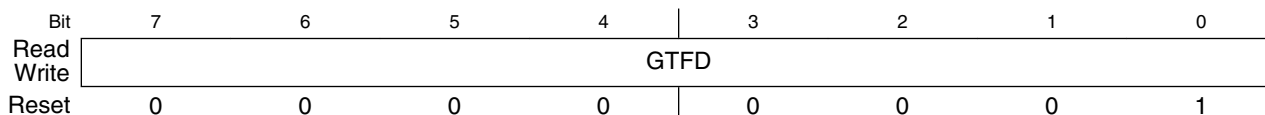
#### UARTx\_WN7816 field descriptions

Field	Description
GTN	Guard Band N  Defines a parameter used in the calculation of GT, CGT, and BGT counters. The value represents an integer number between 0 and 255. See <a href="#">Wait time and guard time parameters</a> .

### 36.4.18 UART 7816 Wait FD Register (UARTx\_WF7816)

The WF7816 contains parameters that are used in the generation of various counters including GT, CGT, BGT, WT, and BWT. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 1Dh offset = 4006\_C01Dh



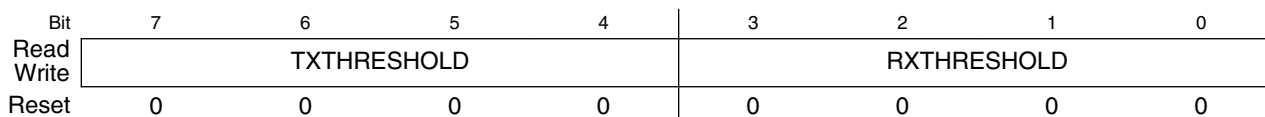
#### UARTx\_WF7816 field descriptions

Field	Description
GTFD	FD Multiplier  Used as another multiplier in the calculation of BWT. This value represents a number between 1 and 255. The value of 0 is invalid. This value is not used in baud rate generation. See <a href="#">Wait time and guard time parameters</a> and <a href="#">Baud rate generation</a> .

### 36.4.19 UART 7816 Error Threshold Register (UARTx\_ET7816)

The ET7816 register contains fields that determine the number of NACKs that must be received or transmitted before the host processor is notified. This register may be read at anytime. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 1Eh offset = 4006\_C01Eh



#### UARTx\_ET7816 field descriptions

Field	Description
7-4 TXTHRESHOLD	Transmit NACK Threshold  The value written to this field indicates the maximum number of failed attempts (NACKs) a transmitted character can have before the host processor is notified. This field is meaningful only when C7816[TTYTYPE] = 0 and C7816[ANACK] = 1. The value read from this field represents the number of consecutive NACKs that have been received since the last successful transmission. This counter saturates at 4'hF and does not wrap around. Regardless of how many NACKs that are received, the UART continues to retransmit indefinitely. This flag only asserts when C7816[TTYTYPE] = 0. For additional information see the IS7816[TXT] field description.  0 TXT asserts on the first NACK that is received. 1 TXT asserts on the second NACK that is received.

Table continues on the next page...

**UARTx\_ET7816 field descriptions (continued)**

Field	Description
RXTHRESHOLD	<p>Receive NACK Threshold</p> <p>The value written to this field indicates the maximum number of consecutive NACKs generated as a result of a parity error or receiver buffer overruns before the host processor is notified. After the counter exceeds that value in the field, the IS7816[RXT] is asserted. This field is meaningful only when C7816[TTYTYPE] = 0. The value read from this field represents the number of consecutive NACKs that have been transmitted since the last successful reception. This counter saturates at 4'hF and does not wrap around. Regardless of the number of NACKs sent, the UART continues to receive valid packets indefinitely. For additional information, see IS7816[RXT] field description.</p>

**36.4.20 UART 7816 Transmit Length Register (UARTx\_TL7816)**

The TL7816 register is used to indicate the number of characters contained in the block being transmitted. This register is used only when C7816[TTYTYPE] = 1. This register may be read at anytime. This register must be written only when C2[TE] is not enabled.

Address: 4006\_C000h base + 1Fh offset = 4006\_C01Fh

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

**UARTx\_TL7816 field descriptions**

Field	Description
TLEN	<p>Transmit Length</p> <p>This value plus four indicates the number of characters contained in the block being transmitted. This register is automatically decremented by 1 for each character in the information field portion of the block. Additionally, this register is automatically decremented by 1 for the first character of a CRC in the epilogue field. Therefore, this register must be programmed with the number of bytes in the data packet if an LRC is being transmitted, and the number of bytes + 1 if a CRC is being transmitted. This register is not decremented for characters that are assumed to be part of the Prologue field, that is, the first three characters transmitted in a block, or the LRC or last CRC character in the Epilogue field, that is, the last character transmitted. This field must be programmed or adjusted only when C2[TE] is cleared.</p>

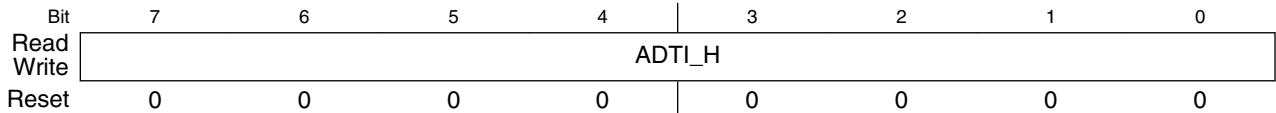
**36.4.21 UART 7816 ATR Duration Timer Register A (UARTx\_AP7816A\_T0)**

The AP7816A\_T0 register contains variables used in the generation of the ATR Duration Timer. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set, except when writing 0 to clear the ADT Counter.

**NOTE**

The ADT Counter starts counting on detection of the complete TS Character. It must be noted that by this time, exactly 10 ETUs have elapsed since the start bit of the TS character. The user must take this into account while programming this register.

Address: 4006\_C000h base + 3Ah offset = 4006\_C03Ah



**UARTx\_AP7816A\_T0 field descriptions**

Field	Description
ADTI_H	<p>ATR Duration Time Integer High (C7816[TTYPE] = 0)</p> <p>Used to calculate the value used for the ADT Counter. This register field provides the most significant byte of the 16 bit ATR Duration Time Integer field ADTI formed by {AP7816A_T0[ADTI_H], AP7816B_T0[ADTI_L]}. Programming a value of ADTI = 0 disables the ADT counter. This value is used only when C7816[TTYPE] = 0. See <a href="#">ATR Duration Time Counter</a>.</p>

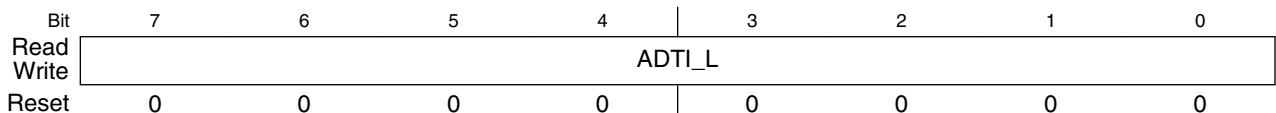
**36.4.22 UART 7816 ATR Duration Timer Register B (UARTx\_AP7816B\_T0)**

The AP7816B\_T0 register contains variables used in the generation of the ATR Duration Timer. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set, except when writing 0 to clear the ADT Counter.

**NOTE**

The ADT Counter starts counting on detection of the complete TS Character. It must be noted that by this time, exactly 10 ETUs have elapsed since the start bit of the TS character. The user must take this into account while programming this register.

Address: 4006\_C000h base + 3Bh offset = 4006\_C03Bh



**UARTx\_AP7816B\_T0 field descriptions**

Field	Description
ADTI_L	ATR Duration Time Integer Low (C7816[TTYPE] = 0)

**UARTx\_AP7816B\_T0 field descriptions (continued)**

Field	Description
	Used to calculate the value used for the ADT counter. This register field provides the least significant byte of the 16 bit ATR Duration Time Integer field ADTI formed by {AP7816A_T0[ADTI_H], AP7816B_T0[ADTI_L]}. Programming a value of ADTI = 0 disables the ADT counter. This value is used only when C7816[TTYPE] = 0. See <a href="#">ATR Duration Time Counter</a> .

**36.4.23 UART 7816 Wait Parameter Register A (UARTx\_WP7816A\_T0)**

The WP7816A\_T0 register contains constants used in the generation of various wait time counters. To save register space, this register is used differently when C7816[TTYPE] = 0 and C7816[TTYPE] = 1. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 3Ch offset = 4006\_C03Ch

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

**UARTx\_WP7816A\_T0 field descriptions**

Field	Description
WI_H	Wait Time Integer High (C7816[TTYPE] = 0)  Used to calculate the value used for the WT counter. This register field provides the most significant byte of the 16 bit Wait Time Integer field WI formed by {WP7816A_T0[WI_H], WP7816B_T0[WI_L]}. The value of WI = 0 is invalid and must not be programmed. This value is used only when C7816[TTYPE] = 0. See <a href="#">Wait time and guard time parameters</a> .

**36.4.24 UART 7816 Wait Parameter Register A (UARTx\_WP7816A\_T1)**

The WP7816A\_T1 register contains constants used in the generation of various wait time counters. To save register space, this register is used differently when C7816[TTYPE] = 0 and C7816[TTYPE] = 1. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 3Ch offset = 4006\_C03Ch

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

### UARTx\_WP7816A\_T1 field descriptions

Field	Description
BWI_H	Block Wait Time Integer High (C7816[TTYPER] = 1)  Used to calculate the value used for the BWT counter. This register field provides the most significant byte of the 16 bit Block Wait Time Integer field BWI formed by {WP7816A_T1[BWI_H], WP7816B_T1[BWI_L]}. The value of BWI = 0 is invalid and should not be programmed. This value is used only when C7816[TTYPER] = 1. See <a href="#">Wait time and guard time parameters</a> .

### 36.4.25 UART 7816 Wait Parameter Register B (UARTx\_WP7816B\_T0)

The WP7816B\_T0 register contains constants used in the generation of various wait time counters. To save register space, this register is used differently when C7816[TTYPER] = 0 and C7816[TTYPER] = 1. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 3Dh offset = 4006\_C03Dh

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

### UARTx\_WP7816B\_T0 field descriptions

Field	Description
WI_L	Wait Time Integer Low (C7816[TTYPER] = 0)  Used to calculate the value used for the WT counter. This register field provides the least significant byte of the 16 bit Wait Time Integer field WI formed by {WP7816A_T0[WI_H], WP7816B_T0[WI_L]}. The value of WI = 0 is invalid and must not be programmed. This value is used only when C7816[TTYPER] = 0. See <a href="#">Wait time and guard time parameters</a> .

### 36.4.26 UART 7816 Wait Parameter Register B (UARTx\_WP7816B\_T1)

The WP7816B\_T1 register contains constants used in the generation of various wait time counters. To save register space, this register is used differently when C7816[TTYPER] = 0 and C7816[TTYPER] = 1. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 3Dh offset = 4006\_C03Dh

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



### UARTx\_WP7816B\_T1 field descriptions

Field	Description
BWI_L	Block Wait Time Integer Low (C7816[TTYTYPE] = 1)  Used to calculate the value used for the BWT counter. This register field provides the least significant byte of the 16 bit Block Wait Time Integer field BWI formed by {WP7816A_T1[BWI_H], WP7816B_T1[BWI_L]}. The value of BWI = 0 is invalid and should not be programmed. This value is used only when C7816[TTYTYPE] = 1. See <a href="#">Wait time and guard time parameters</a> .

### 36.4.27 UART 7816 Wait and Guard Parameter Register (UARTx\_WGP7816\_T1)

The WGP7816\_T1 register contains constants used in the generation of various wait and guard timer counters. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 3Eh offset = 4006\_C03Eh

Bit	7	6	5	4	3	2	1	0
Read	CW11				BGI			
Write								
Reset	0	0	0	0	0	1	1	0

### UARTx\_WGP7816\_T1 field descriptions

Field	Description
7-4 CW11	Character Wait Time Integer 1 (C7816[TTYTYPE] = 1)  Used to calculate the value used for the CWT counter. It represents a value between 0 and 15. This value is used only when C7816[TTYTYPE] = 1. See <a href="#">Wait time and guard time parameters</a> .
BGI	Block Guard Time Integer (C7816[TTYTYPE] = 1)  Used to calculate the value used for the BGT counter. It represent a value between 0 and 15. This value is used only when C7816[TTYTYPE] = 1. See <a href="#">Wait time and guard time parameters</a> .

### 36.4.28 UART 7816 Wait Parameter Register C (UARTx\_WP7816C\_T1)

The WP7816C\_T1 register contains constants used in the generation of various wait timer counters. This register may be read at any time. This register must be written to only when C7816[ISO\_7816E] is not set.

Address: 4006\_C000h base + 3Fh offset = 4006\_C03Fh

Bit	7	6	5	4	3	2	1	0
Read	0				CW12			
Write								
Reset	0	0	0	0	1	0	1	1

### UARTx\_WP7816C\_T1 field descriptions

Field	Description
7-5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CWI2	Character Wait Time Integer 2 (C7816[TTYPE] = 1)  Used to calculate the value used for the CWT counter. It represents a value between 0 and 31. This value is used only when C7816[TTYPE] = 1. See <a href="#">Wait time and guard time parameters</a> .

## 36.5 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.

## 36.5.1 Transmitter

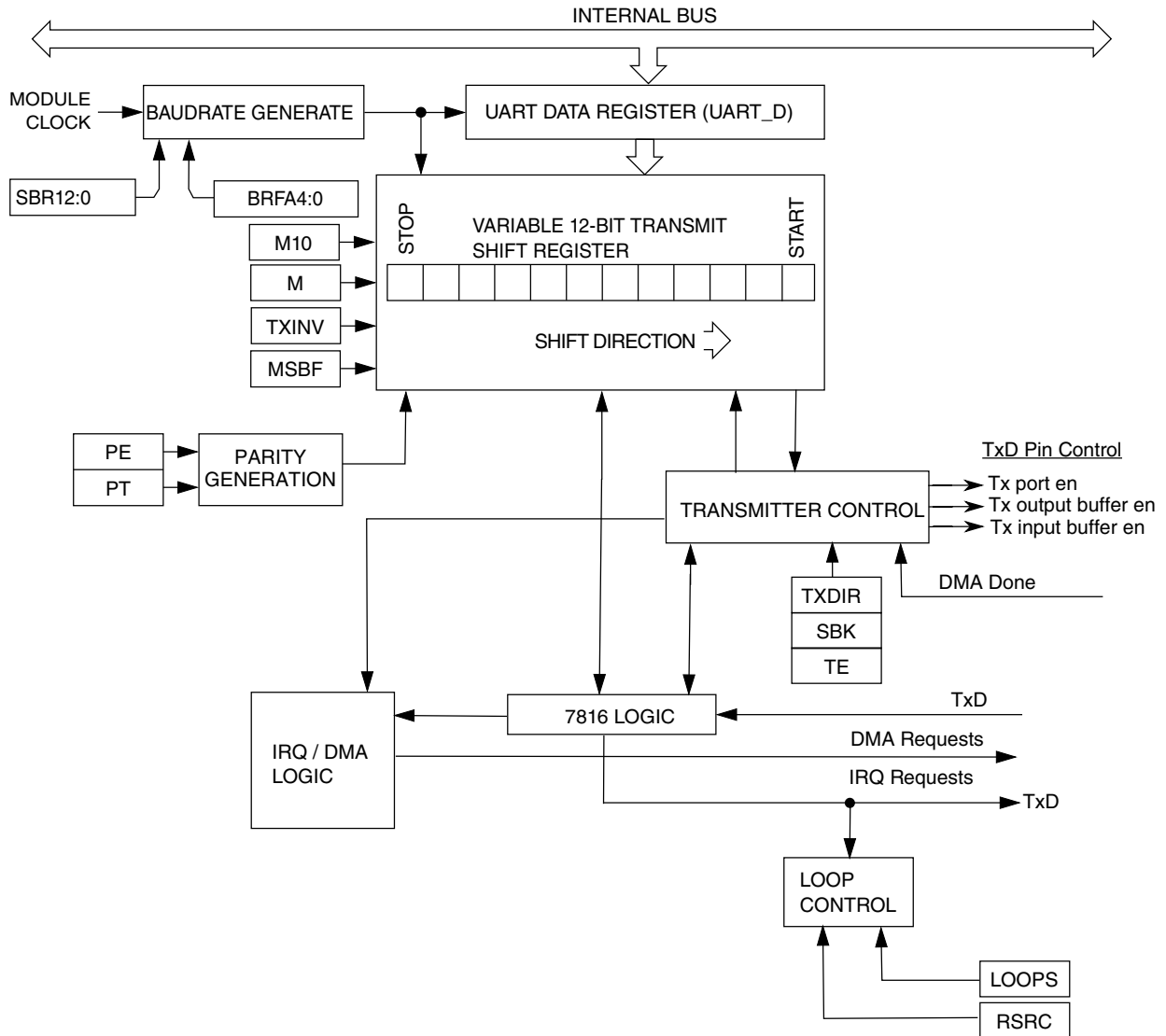


Figure 36-1. Transmitter Block Diagram

### 36.5.1.1 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).

### 36.5.1.2 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.

### 36.5.1.3 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]) when transmit buffer is empty.. 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.

When C7816[ISO\_7816E] = 1, setting C2[TE] does not result in a preamble being generated. The transmitter starts transmitting as soon as the corresponding guard time expires. When C7816[TTYTYPE] = 0, the value in GT is used. When C7816[TTYTYPE] = 1, the value in BGT is used, because C2[TE] will remain asserted until the end of the block transfer. C2[TE] is automatically cleared when C7816[TTYTYPE] = 1 and the block being transmitted has completed. When C7816[TTYTYPE] = 0, the transmitter listens for a NACK indication. If no NACK is received, it is assumed that the character was correctly

received. If a NACK is received, the transmitter resends the data, assuming that the number of retries for that character, that is, the number of NACKs received, is less than or equal to the value in ET7816[TXTHRESHOLD].

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.

### 36.5.1.4 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 36-3. 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. Break bits are not supported when C7816[ISO\_7816E] is set/enabled.

**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.

**36.5.1.5 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]. When C7816[ISO\_7816E] is set/enabled, idle characters are not sent or detected. When data is not being transmitted, the data I/O line is in an inactive state.

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.

**36.5.2 Receiver**

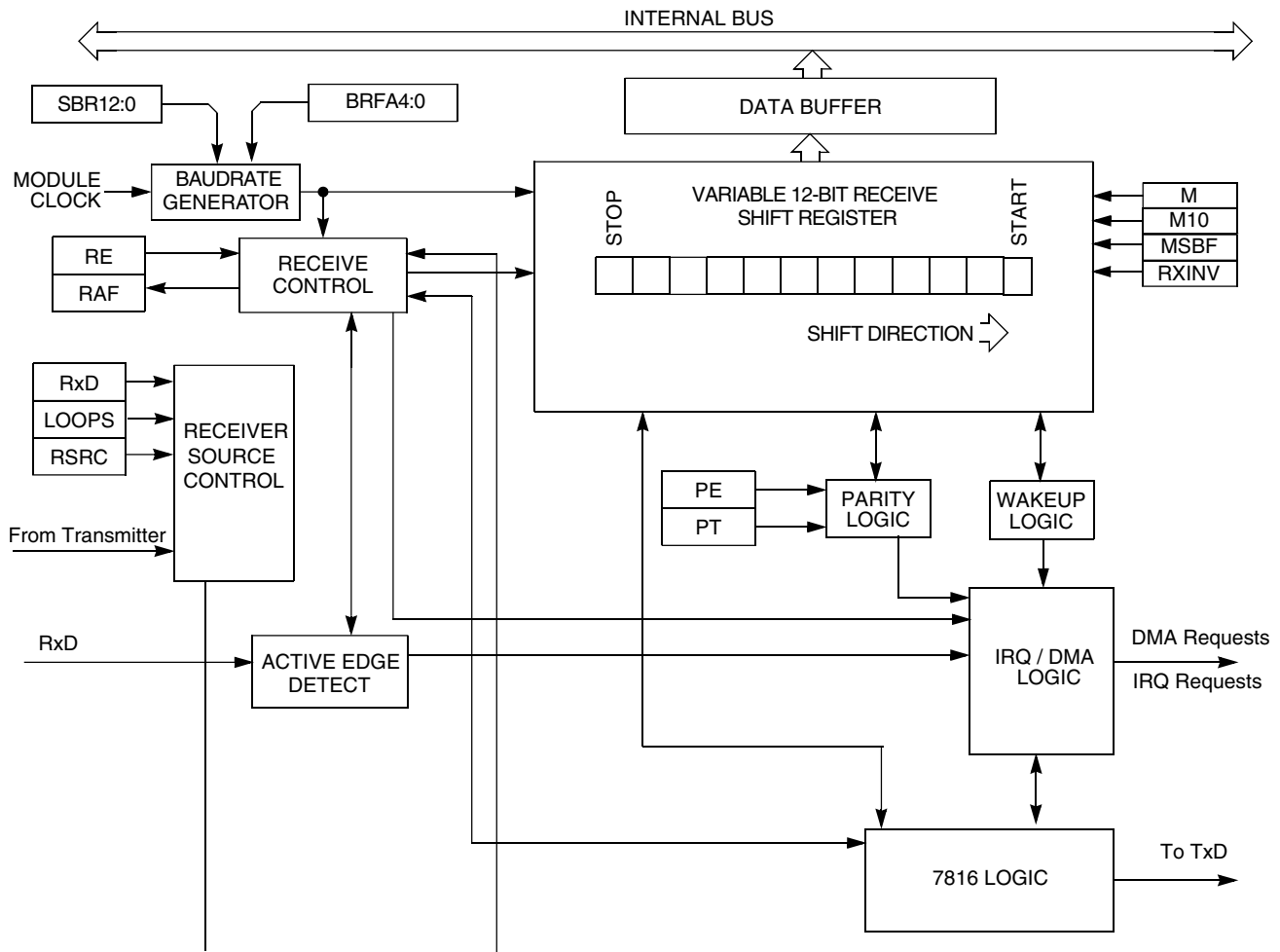


Figure 36-2. UART receiver block diagram

### 36.5.2.1 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).

### 36.5.2.2 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].

### 36.5.2.3 Character reception

During UART reception, the receive shift register shifts a frame in from the unsynchronized receiver input signal. After a complete frame shifts into the receive shift register, the data portion of the frame transfers to the UART receive buffer. The receive data buffer is accessible via the D and C3[T8] registers. S1[RDRF] is set if the receive buffer is full. If the C2[RIE] is also set, RDRF generates an RDRF interrupt request. Alternatively, by programming C5[RDMAS], a DMA request can be generated.

When C7816[ISO\_7816E] is set/enabled and C7816[TTYTYPE] = 0, character reception operates slightly differently. Upon receipt of the parity bit, the validity of the parity bit is checked. If C7816[ANACK] is set and the parity check fails, or if INIT and the received character is not a valid initial character, then a NACK is sent by the receiver. If the number of consecutive receive errors exceeds the threshold set by ET7816[RXTHRESHOLD], then IS7816[RXT] is set and an interrupt generated if IE7816[RXTE] is set. If an error is detected due to parity or an invalid initial character, the data is not transferred from the receive shift register to the receive buffer. Instead, the data is overwritten by the next incoming data.

When the C7816[ISO\_7816E] is set/enabled, C7816[ONACK] is set/enabled, and the received character results in the receive buffer overflowing, a NACK is issued by the receiver. Additionally, S1[OR] is set and an interrupt is issued if required, and the data in the shift register is discarded.

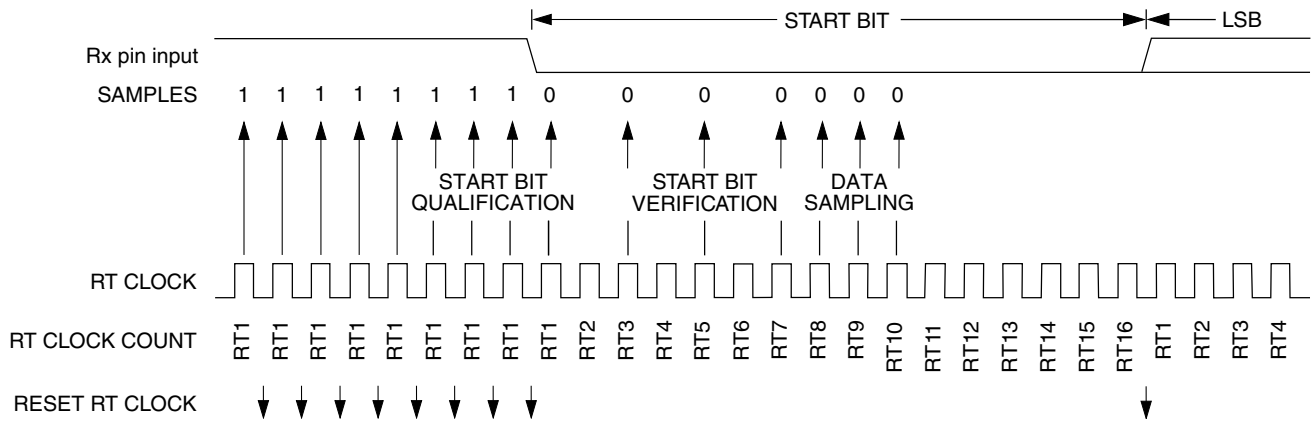
### 36.5.2.4 Data sampling

The receiver samples the unsynchronized receiver input signal at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock (see the following figure) is re-synchronized:

- After every start bit.
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0).

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.





**Figure 36-3. Receiver data sampling**

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7 when C7816[ISO\_7816E] is cleared/disabled and RT8, RT9 and RT10 when C7816[ISO\_7816E] is set/enabled. The following table summarizes the results of the start bit verification samples.

**Table 36-4. Start bit verification**

RT3, RT5, and RT7 samples RT8, RT9, RT10 samples when 7816E	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 36-5. 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

*Table continues on the next page...*

**Table 36-5. Data bit recovery (continued)**

RT8, RT9, and RT10 samples	Data bit determination	Noise flag
100	0	1
101	1	1
110	1	1
111	1	0

**Note**

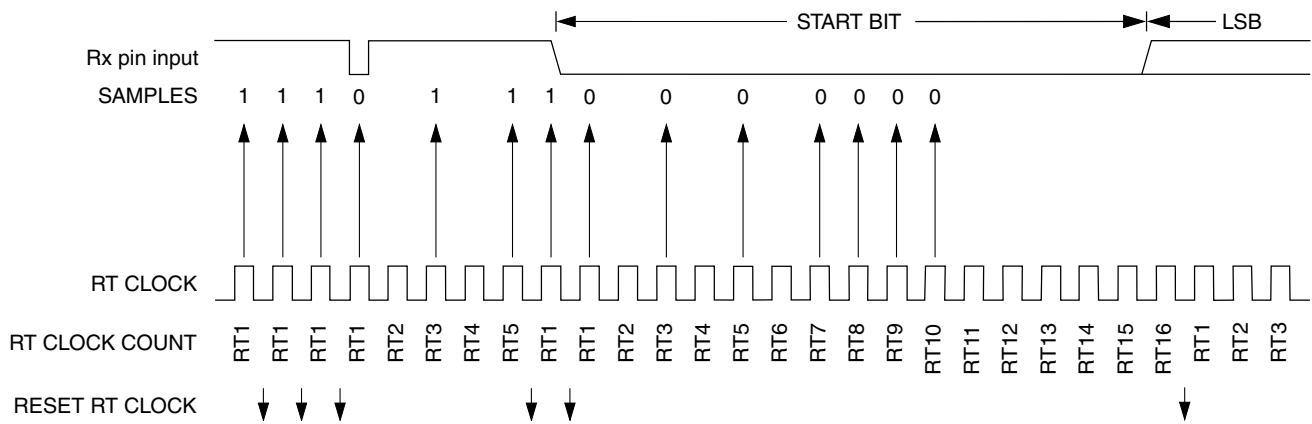
The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (S1[NF]) is set and the receiver assumes that the bit is a start bit (logic 0). With the exception of when C7816[ISO\_7816E] is set/enabled, where the values of RT8, RT9 and RT10 exclusively determine if a start bit exists.

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. In the event that C7816[ISO\_7816E] is set/enabled and C7816[TTYPE] = 0, verification of a stop bit does not take place. Rather, starting with RT8 the receiver transmits a NACK as programmed until time RT9 of the following time period. Framing Error detection is not supported when C7816[ISO\_7816E] is set/enabled.

**Table 36-6. Stop bit recovery**

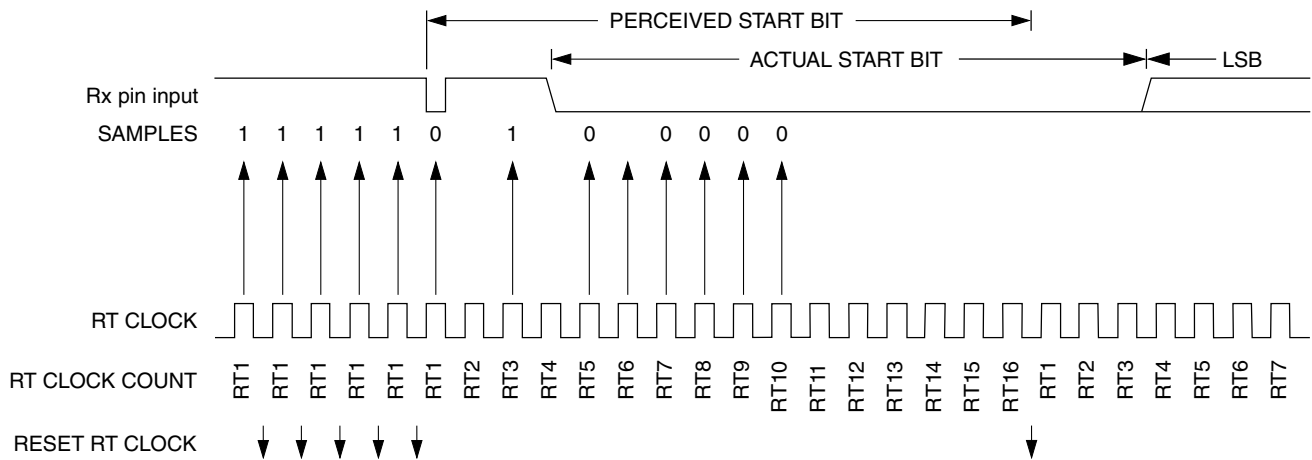
RT8, RT9, and RT10 samples	Framing error flag	Noise flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

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. In this example C7816[ISO\_7816E] = 0. 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 36-4. Start bit search example 1 (C7816[ISO\_7816E] = 0)**

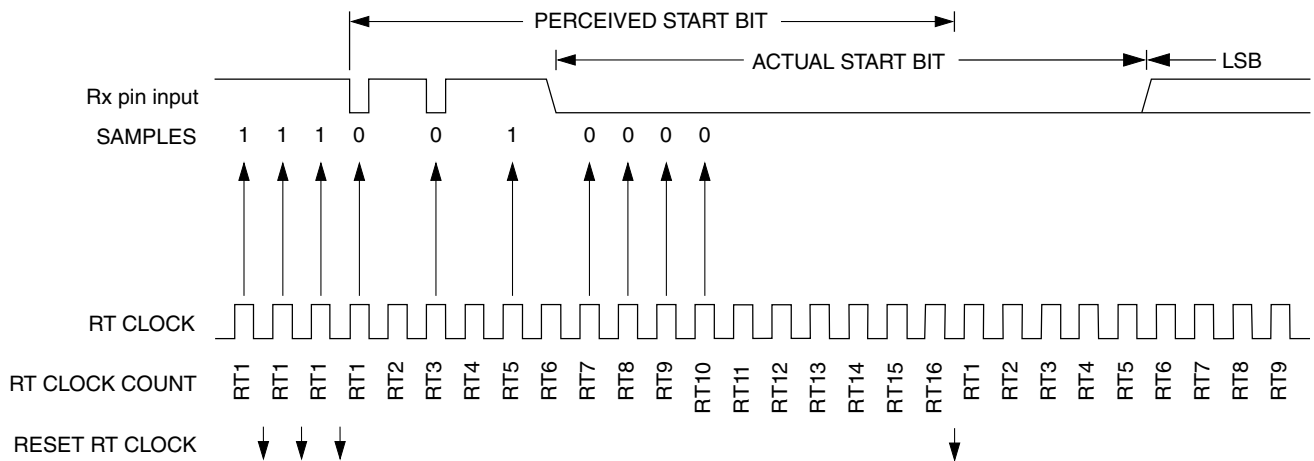
In the following figure, verification sample at RT3 is high. In this example C7816[ISO\_7816E] = 0. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.



**Figure 36-5. Start bit search example 2 (C7816[ISO\_7816E] = 0)**

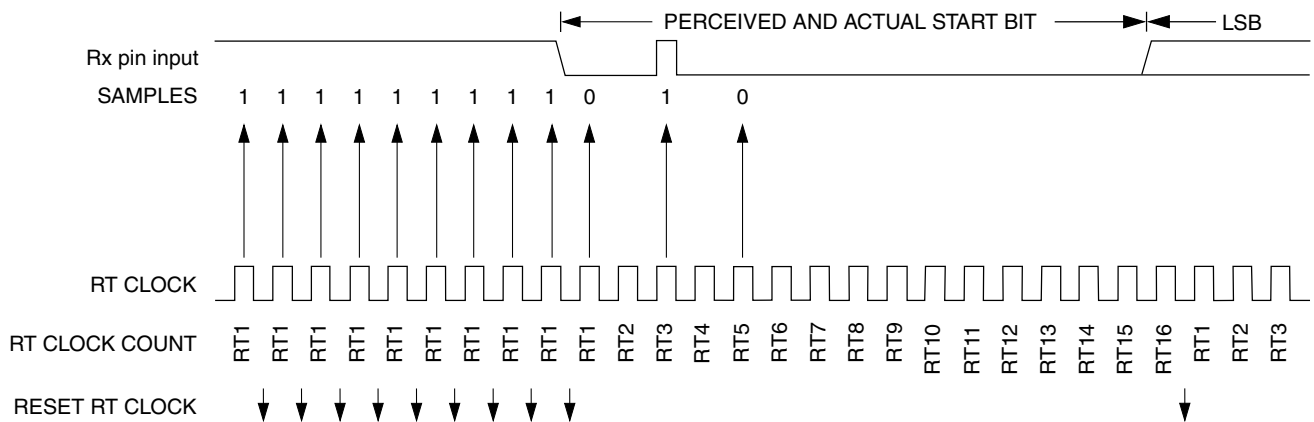
In the following figure, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. In this example C7816[ISO\_7816E] = 0. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

## Functional description



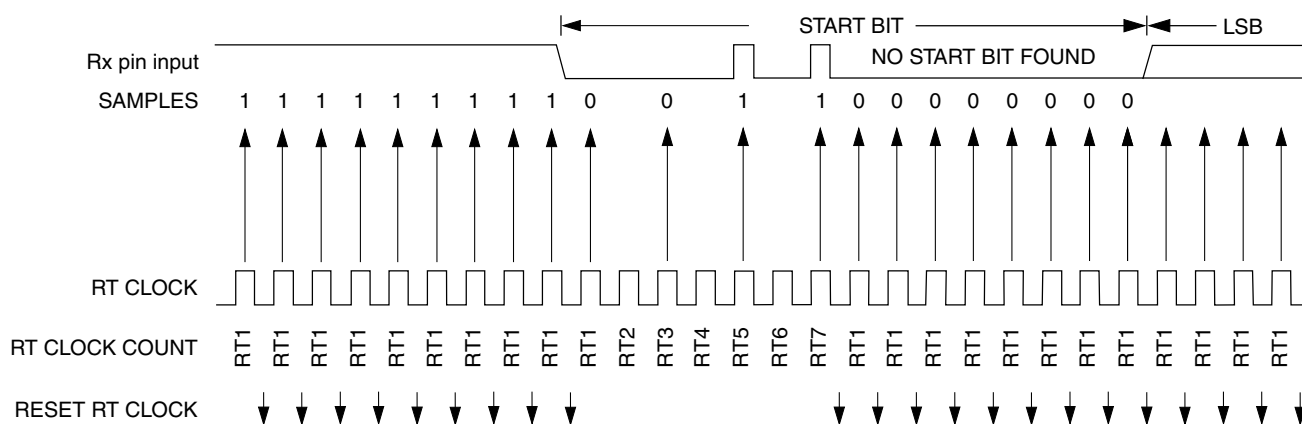
**Figure 36-6. Start bit search example 3 (C7816[ISO\_7816E] = 0)**

The following figure shows the effect of noise early in the start bit time. In this example C7816[ISO\_7816E] = 0. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.



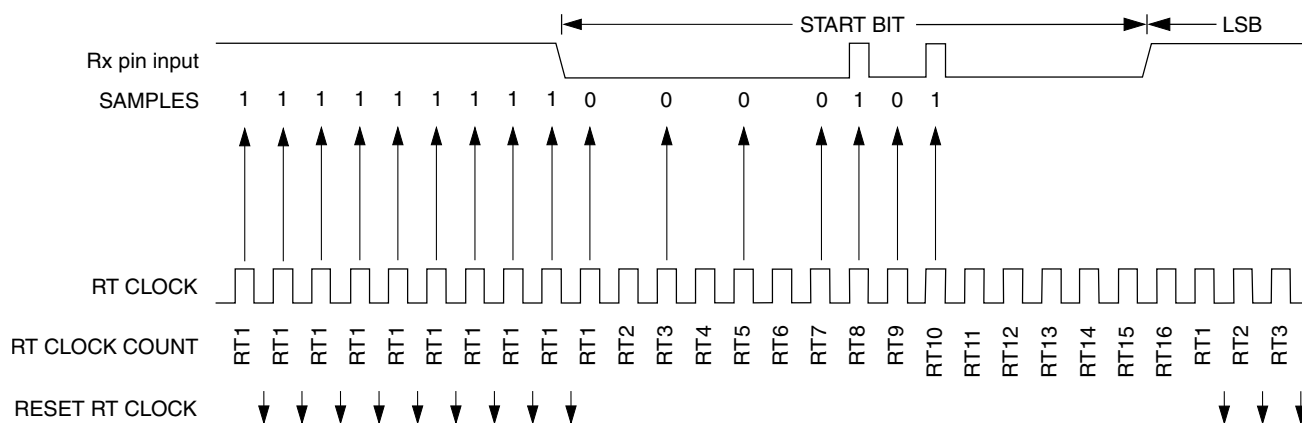
**Figure 36-7. Start bit search example 4 (C7816[ISO\_7816E] = 0)**

The following figure shows a burst of noise near the beginning of the start bit that resets the RT clock. In this example C7816[ISO\_7816E] = 0. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.



**Figure 36-8. Start bit search example 5 (C7816[ISO\_7816E] = 0)**

In the following figure, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. In this example C7816[ISO\_7816E] = 0. This sets the noise flag but does not reset the RT clock. In start bits only, the RT8, RT9, and RT10 data samples are ignored. In this example, if C7816[ISO\_7816E] = 1 then a start bit would not have been detected at all since at least two of the three samples (RT8, RT9, RT10) were high.



**Figure 36-9. Start bit search example 6**

### 36.5.2.5 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]. 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. Framing errors are not supported when C7816[ISO7816E] is set/enabled. However, if S1[FE] is set, data will not be received when C7816[ISO7816E] is set.

### 36.5.2.6 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 UART break character detection threshold depends on C1[M], C1[PE] and C4[M10]. See the following table.

**Table 36-7. Receive break character detection threshold**

M	M10	PE	Threshold (bits)
0	—	—	10
1	0	—	11
1	1	1	12

Break characters are not detected or supported when C7816[ISO\_7816E] is set/enabled.

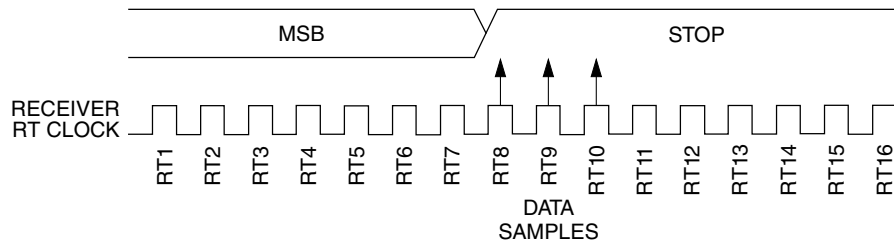
### 36.5.2.7 Baud rate tolerance

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.

#### 36.5.2.7.1 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 36-10. 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).

With the misaligned character shown in the [Figure 36-10](#), 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).

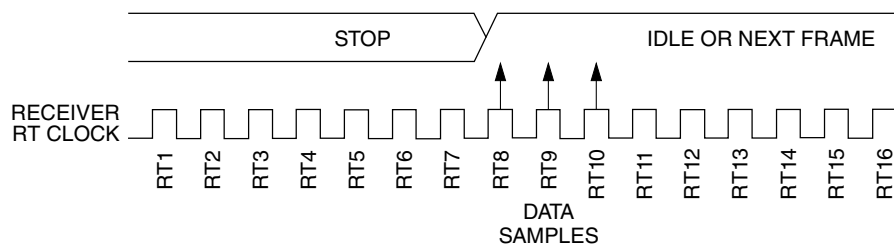
With the misaligned character shown in the [Figure 36-10](#), the receiver counts 170 RT cycles at the point when the count of the transmitting device is 163 RT cycles (10 bit times  $\times$  16 RT cycles + 3 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$((170 - 163) \div 170) \times 100 = 4.12\%$$

### 36.5.2.7.2 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 36-11. Fast data**

For an 8-bit data character, data sampling of the stop bit takes the receiver 154 RT cycles (9 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the [Figure 36-11](#), the receiver counts 154 RT cycles at the point when the count of the transmitting device is 160 RT cycles (10 bit times × 16 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

$$((154 - 160) \div 154) \times 100 = 3.90\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 170 RT cycles (10 bit times × 16 RT cycles + 10 RT cycles).

With the misaligned character shown in the [Figure 36-11](#), the receiver counts 170 RT cycles at the point when the count of the transmitting device is 176 RT cycles (11 bit times × 16 RT cycles).

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$((170 - 176) \div 170) \times 100 = 3.53\%$$

### **36.5.2.8 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.

Receiver wakeup is not supported when C7816[ISO\_7816E] is set/enabled because multi-receiver systems are not allowed.

#### **36.5.2.8.1 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.

Idle input line wakeup is not supported when C7816[ISO\_7816E] is set/enabled.

### 36.5.2.8.2 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.

Address mark wakeup is not supported when C7816[ISO\_7816E] is set/enabled.

### 36.5.2.8.3 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.

Address match operation is not supported when C7816[ISO\_7816E] is set/enabled.

### 36.5.3 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 36-8](#) 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 36-8. 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

*Table continues on the next page...*

**Table 36-8. Baud rates (example: module clock = 10.2 MHz) (continued)**

Bits SBR (decimal)	Bits BRFA	BRFD value	Receiver clock (Hz)	Transmitter clock (Hz)	Target Baud rate	Error (%)
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 36-9. Baud rate fine adjust**

BRFA	Baud Rate Fractional Divisor (BRFD)
0 0 0 0 0	$0/32 = 0$
0 0 0 0 1	$1/32 = 0.03125$
0 0 0 1 0	$2/32 = 0.0625$
0 0 0 1 1	$3/32 = 0.09375$
0 0 1 0 0	$4/32 = 0.125$
0 0 1 0 1	$5/32 = 0.15625$
0 0 1 1 0	$6/32 = 0.1875$
0 0 1 1 1	$7/32 = 0.21875$
0 1 0 0 0	$8/32 = 0.25$
0 1 0 0 1	$9/32 = 0.28125$
0 1 0 1 0	$10/32 = 0.3125$
0 1 0 1 1	$11/32 = 0.34375$
0 1 1 0 0	$12/32 = 0.375$
0 1 1 0 1	$13/32 = 0.40625$
0 1 1 1 0	$14/32 = 0.4375$
0 1 1 1 1	$15/32 = 0.46875$
1 0 0 0 0	$16/32 = 0.5$
1 0 0 0 1	$17/32 = 0.53125$
1 0 0 1 0	$18/32 = 0.5625$
1 0 0 1 1	$19/32 = 0.59375$
1 0 1 0 0	$20/32 = 0.625$
1 0 1 0 1	$21/32 = 0.65625$

Table continues on the next page...

**Table 36-9. Baud rate fine adjust (continued)**

BRFA	Baud Rate Fractional Divisor (BRFD)
1 0 1 1 0	$22/32 = 0.6875$
1 0 1 1 1	$23/32 = 0.71875$
1 1 0 0 0	$24/32 = 0.75$
1 1 0 0 1	$25/32 = 0.78125$
1 1 0 1 0	$26/32 = 0.8125$
1 1 0 1 1	$27/32 = 0.84375$
1 1 1 0 0	$28/32 = 0.875$
1 1 1 0 1	$29/32 = 0.90625$
1 1 1 1 0	$30/32 = 0.9375$
1 1 1 1 1	$31/32 = 0.96875$

### 36.5.4 Data format (non ISO-7816)

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].

#### 36.5.4.1 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 36-10. 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](#).
2. The address bit identifies the frame as an address character. See [Receiver wakeup](#).

### 36.5.4.2 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 36-11. 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>				
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>1</sup>	1	1

1. 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.

### 36.5.4.3 Timing examples

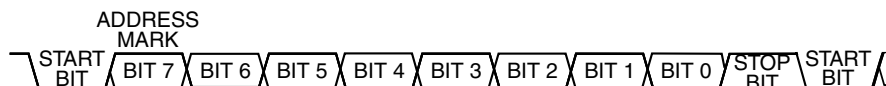
Timing examples of these configurations in the NRZ mark/space data format are illustrated in the following figures. The timing examples show all of the configurations in the following sub-sections along with the LSB and MSB first variations.

### 36.5.4.3.1 Eight-bit format with parity disabled

The most significant bit can be used for address mark wakeup.

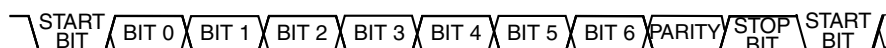


**Figure 36-12. Eight bits of data with LSB first**



**Figure 36-13. Eight bits of data with MSB first**

### 36.5.4.3.2 Eight-bit format with parity enabled



**Figure 36-14. Seven bits of data with LSB first and parity**



**Figure 36-15. Seven bits of data with MSB first and parity**

### 36.5.4.3.3 Nine-bit format with parity disabled

The most significant bit can be used for address mark wakeup.

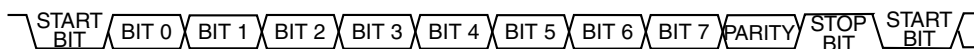


**Figure 36-16. Nine bits of data with LSB first**

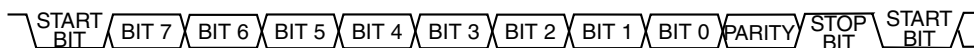


**Figure 36-17. Nine bits of data with MSB first**

### 36.5.4.3.4 Nine-bit format with parity enabled



**Figure 36-18. Eight bits of data with LSB first and parity**



**Figure 36-19. Eight bits of data with MSB first and parity**

### 36.5.4.3.5 Non-memory mapped tenth bit for parity

The most significant memory-mapped bit can be used for address mark wakeup.



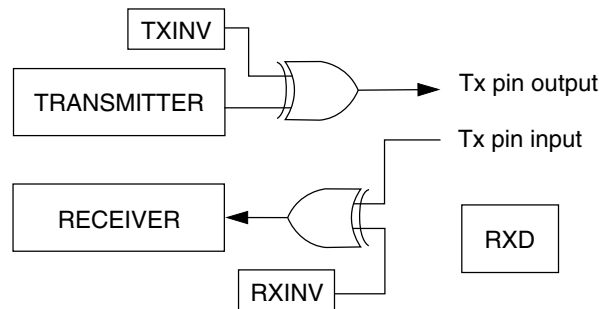
**Figure 36-20. Nine bits of data with LSB first and parity**



**Figure 36-21. Nine bits of data with MSB first and parity**

## 36.5.5 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.

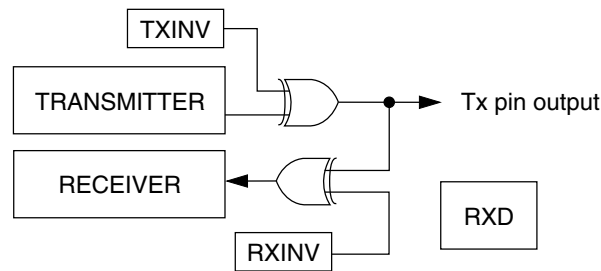


**Figure 36-22. Single-wire operation (C1[LOOPS] = 1, C1[RSRC] = 1)**

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). When C7816[ISO\_7816EN] is set, it is not required that both C2[TE] and C2[RE] are set.

## 36.5.6 Loop operation

In loop operation, the transmitter output goes to the receiver input. The unsynchronized receiver input signal is disconnected from the UART.



**Figure 36-23. 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). When C7816[ISO\_7816EN] is set, it is not required that both C2[TE] and C2[RE] are set.

### 36.5.7 ISO-7816/smartcard support

The UART provides mechanisms to support the ISO-7816 protocol that is commonly used to interface with smartcards. The ISO-7816 protocol is an NRZ, single wire, half-duplex interface. The TxD pin is used in open-drain mode because the data signal is used for both transmitting and receiving. There are multiple subprotocols within the ISO-7816 standard. The UART supports both T = 0 and T = 1 protocols. The module also provides for automated initial character detection and configuration, which allows for support of both direct convention and inverse convention data formats. A variety of interrupts specific to 7816 are provided in addition to the general interrupts to assist software. Additionally, the module is able to provide automated NACK responses and has programmed automated retransmission of failed packets. An assortment of programmable timeouts and guard band times are also supported.

The term elemental time unit (ETU) is frequently used in the context of ISO-7816. This concept is used to relate the frequency that the system (UART) is running at and the frequency that data is being transmitted and received. One ETU represents the time it takes to transmit or receive a single bit. For example, a standard 7816 packet, excluding any guard time or NACK elements is 10 ETUs (start bit, 8 data bits, and a parity bit). Guard times and wait times are also measured in ETUs.,

#### NOTE

The ISO-7816 specification may have certain configuration options that are reserved. To maintain maximum flexibility to support future 7816 enhancements or devices that may not strictly conform to the specification, the UART does not prevent those options being used. Further, the UART may



provide configuration options that exceed the flexibility of options explicitly allowed by the 7816 specification. Failure to correctly configure the UART may result in unexpected behavior or incompatibility with the ISO-7816 specification.

### 36.5.7.1 Initial characters

In ISO-7816 with T = 0 mode, the UART can be configured to use C7816[INIT] to detect the next valid initial character, referred to by the ISO-7816 specifically as a TS character. When the initial character is detected, the UART provides the host processor with an interrupt if IE7816[INITDE] is set. Additionally, the UART will alter S2[MSBF], C3[TXINV], and S2[RXINV] automatically, based on the initial character. The corresponding initial character and resulting register settings are listed in the following table.

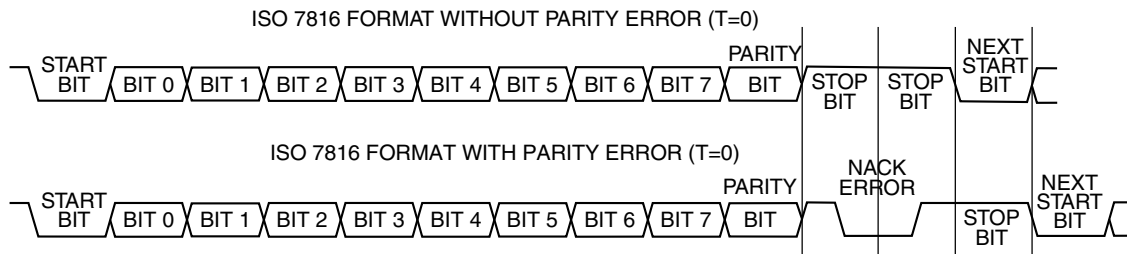
**Table 36-12. Initial character automated settings**

Initial character (bit 1-10)	Initial character (hex)	MSBF	TXINV	RXINV
LHHL LLL LLH inverse convention	3F	1	1	1
LHHL HHH LLH direct convention	3B	0	0	0

S2[MSBF], C3[TXINV], and S2[RXINV] must be reset to their default values before C7816[INIT] is set. Once C7816[INIT] is set, the receiver searches all received data for the first valid initial character. Detecting a Direct Convention Initial Character will cause no change to S2[MSBF], C3[TXINV], and S2[RXINV], while detecting an Inverse Convention Initial Character will cause these fields to set automatically. All data received, which is not a valid initial character, is ignored and all flags resulting from the invalid data are blocked from asserting. If C7816[ANACK] is set, a NACK is returned for invalid received initial characters and an RXT interrupt is generated as programmed.

### 36.5.7.2 Protocol T = 0

When T = 0 protocol is selected, a relatively complex error detection scheme is used. Data characters are formatted as illustrated in the following figure. This scheme is also used for answer to reset and Peripheral Pin Select (PPS) formats.



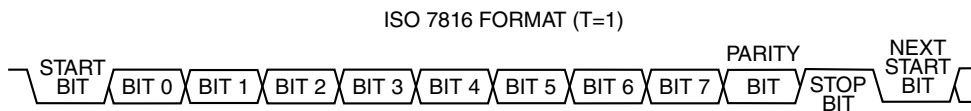
**Figure 36-24. ISO-7816 T = 0 data format**

As with other protocols supported by the UART, the data character includes a start bit. However, in this case, there are two stop bits rather than the typical single stop bit. In addition to a standard even parity check, the receiver has the ability to generate and return a NACK during the second half of the first stop bit period. The NACK must be at least one time period (ETU) in length and no more than two time periods (ETU) in length. The transmitter must wait for at least two time units (ETU) after detection of the error signal before attempting to retransmit the character.

It is assumed that the UART and the device (smartcard) know in advance which device is receiving and which is transmitting. No special mechanism is supplied by the UART to control receive and transmit in the mode other than C2[TE] and C2[RE]. Initial Character Detect feature is also supported in this mode.

### 36.5.7.3 Protocol T = 1

When T = 1 protocol is selected, the NACK error detection scheme is not used. Rather, the parity bit is used on a character basis and a CRC or LRC is used on the block basis, that is, for each group of characters. In this mode, the data format allows for a single stop bit although additional inactive bit periods may be present between the stop bit and the next start bit. Data characters are formatted as illustrated in the following figure.



**Figure 36-25. ISO 7816 T=1 data format**

The smallest data unit that is transferred is a block. A block is made up of several data characters and may vary in size depending on the block type. The UART does not provide a mechanism to decode the block type. As part of the block, an LRC or CRC is included. The UART does not calculate the CRC or LRC for transmitted blocks, nor does it verify the validity of the CRC or LRC for received blocks. The 7816 protocol requires that the initiator and the smartcard (device) takes alternate turns in transmitting and receiving blocks. When the UART detects that the last character in a block has been transmitted it will automatically clear C2[TE], C3[TXDIR] and enter receive mode.

Therefore, the software must program the transmit buffer with the next data to be transmitted and then enable C2[TE] and set C3[TXDIR], once the software has determined that the last character of the received block has been received. The UART detects that the last character of the transmit block has been sent when TL7816[TLEN] = 0 and four additional characters have been sent. The four additional characters are made up of three prior to TL7816[TLEN] decrementing (prologue) and one after TL7816[TLEN] = 0, the final character of the epilogue.

#### 36.5.7.4 Wait time and guard time parameters

The ISO-7816 specification defines several wait time and guard time parameters. The UART allows for flexible configuration and violation detection of these settings. On reset, the wait time (IS7816[WT]) defaults to 9600 ETUs and guard time (GT) to 12 ETUs. These values are controlled by parameters in the WP7816, WN7816, and WF7816 registers. Additionally, the value of C7816[TTYPER] also factors into the calculation. The formulae used to calculate the number ETUs for each wait time and guard time value are shown in [Table 36-13](#).

Wait time (WT) is defined as the maximum allowable time between the leading edge of a character transmitted by the smartcard device and the leading edge of the previous character that was transmitted by the UART or the device. Similarly, character wait time (CWT) is defined as the maximum allowable time between the leading edge of two characters within the same block. Block wait time (BWT) is defined as the maximum time between the leading edge character of the last block received by the smartcard device and the leading edge of the first character transmitted by the smartcard device.

Guard time (GT) is defined as the minimum allowable time between the leading edge of two consecutive characters. Character guard time (CGT) is the minimum allowable time between the leading edges of two consecutive characters in the same direction, that is, transmission or reception. Block guard time (BGT) is the minimum allowable time between the leading edges of two consecutive characters in opposite directions, that is, transmission then reception or reception then transmission.

The GT and WT counters reset whenever C7816[TTYPER] = 1 or C7816[ISO\_7816E] = 0 or a new dataword start bit has been received or transmitted as specified by the counter descriptions. The CWT, CGT, BWT, BGT counters reset whenever C7816[TTYPER] = 0 or C7816[ISO\_7816E] = 0 or a new dataword start bit is received or transmitted as specified by the counter descriptions. When C7816[TTYPER] = 1, some of the counter values require an assumption regarding the first data transferred when the UART first starts. This assumption is required when the 7816E is disabled, when transition from C7816[TTYPER] = 0 to C7816[TTYPER] = 1 or when coming out of reset. In this case, it is assumed that the previous non-existent transfer was a received transfer.

## Functional description

The UART will automatically handle GT, CGT, and BGT such that the UART will not send a packet before the corresponding guard time expiring.

**Table 36-13. Wait and guard time calculations**

Parameter	Reset value [ETU]	C7816[TTYPE] = 0 [ETU]	C7816[TTYPE] = 1 [ETU]
Wait time (WT)	9600	$WI \times 480$	Not used
Character wait time (CWT)	Not used	Not used	$2^{(CWI1)} + CWI2$
Block wait time (BWT)	Not used	Not used	$(11 + (BWI \times 960 \times GTFD)) * (WTX + 1)$
Guard time (GT)	12	<b>GTN not equal to 255</b> $12 + GTN$ <b>GTN equal to 255</b> 12	Not used
Character guard time (CGT)	Not used	Not used	<b>GTN not equal to 255</b> $12 + GTN$ <b>GTN equal to 255</b> 11
Block guard time (BGT)	Not used	Not used	$16 + BGI$

### NOTE

- User must ensure that the Character Wait time (CWT) programmed using the formula above is atleast 12. Values smaller than 12 are invalid and will lead to unexpected CWT interrupts.
- The 16 bit Wait Time integer WI is formed by concatenation of {WP7816A\_T0[WI\_H], WP7816B\_T0[WI\_L]}.
- The 16 bit Block Wait Time integer BWI is formed by concatenation of {WP7816A\_T1[BWI\_H], WP7816B\_T1[BWI\_L]}.

### 36.5.7.5 ATR Duration Time Counter

The ISO-7816 specification defines a specific time (in etus) within which the terminal must receive the ATR (Answer to Reset), failing which the terminal must abort the card session by initiating the deactivation sequence.

UART supports this in hardware via the ATR Duration Time (ATD) Counter which can be programmed using AP7816a\_T0 and AP7816b\_T0 registers. The value loaded into the ADT (ATR Duration Time) counter is given by the concatenation of the register fields as

shown;  $ADT = \{AP7816a\_T0[ADTI\_H], AP7816a\_T0[ADTI\_L]\}$ . This counter begins to count on detection of the TS character which is detected when IS7816[INITD] flag is set. Once the ATR process is completed, the ATD Counter must be disabled by writing 0 to AP7816x\_T0 registers, in order to prevent the false occurrence of the ATD Duration Time interrupt IS7816[ATD]. Note that this feature is only supported in T = 0 mode.

### NOTE

The ADT counter starts counting on detection of the complete TS Character. It must be noted that by this time, exactly 10 ETUs have elapsed since the start bit of the TS character. The user must take this into account while programming AP7816a\_T0 and AP7816b\_T0 registers.

### 36.5.7.6 Baud rate generation

The value in WF7816[GTFD] does not impact the clock frequency. SBR and BRFD are used to generate the clock frequency. This clock frequency is used by the UART only and is not seen by the smartcard device. The transmitter clocks operates at 1/16 the frequency of the receive clock so that the receiver is able to sample the received value 16 times during the ETU.

### 36.5.7.7 UART restrictions in ISO-7816 operation

Due to the flexibility of the UART module, there are several features and interrupts that are not supported while running in ISO-7816 mode. These restrictions are documented within the register field definitions.

## 36.6 Reset

All registers reset to a particular value are indicated in [Memory map and registers](#).

## 36.7 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 36-14. 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	RXEDGIF	RXEDGIE	-
Receiver	OR	ORIE	-
Receiver	NF	NEIE	-
Receiver	FE	FEIE	-
Receiver	PF	PEIE	-
Receiver	WT	WTWE	-
Receiver	CWT	CWTE	-
Receiver	BWT	BWTE	-
Receiver	INITD	INITDE	-
Receiver	TXT	TXTE	-
Receiver	RXT	RXTE	-
Receiver	GTV	GTVE	-

## 36.7.1 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.

### 36.7.1.1 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.

### 36.7.1.2 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.

### 36.7.1.3 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).

## 36.8 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 36-15. 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.

## 36.9 Application information

This section describes the UART application information.

### 36.9.1 ISO-7816 initialization sequence

This section outlines how to program the UART for ISO-7816 operation. Elements such as procedures to power up or power down the smartcard, and when to take those actions, are beyond the scope of this description. To set up the UART for ISO-7816 operation:

1. 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. According to the 7816 specification the initial (default) baud rating setting should be  $F_i = 372$  and  $D_i = 1$  and a maximum frequency of 5 MHz. In other words, the BDH, BDL, and C4 registers should be programmed such that the transmission frequency provided to the smartcard device must be  $1/372$ th of the clock and must not exceed 5 MHz.
2. Write to C1 to configure word length, parity, and other configuration fields (LOOPS, RSRC) and set  $C1[M] = 1$ ,  $C1[PE] = 1$ , and  $C1[PT] = 0$ .
3. Write to set  $S2[RWUID] = 0$
4. Write to set up interrupt enable fields desired ( $C3[ORIE]$ ,  $C3[NEIE]$ ,  $C3[PEIE]$ , and  $C3[FEIE]$ )
5. Write to set  $C4[MAEN1] = 0$  and  $C4[MAEN2] = 0$ .
6. Write to C5 register and configure DMA control register fields as desired for application.
7. Write to set  $C7816[INIT] = 1$ ,  $C7816[TTYTYPE] = 0$ , and  $C7816[ISO_7816E] = 1$ . Program  $C7816[ONACK]$  and  $C7816[ANACK]$  as desired.
8. Write to IE7816 to set interrupt enable parameters as desired.
9. Write to ET7816 and set as desired.
10. Write to set  $C2[ILIE] = 0$ ,  $C2[RE] = 1$ ,  $C2[TE] = 1$ ,  $C2[RWU] = 0$ , and  $C2[SBK] = 0$ . Set up interrupt enables  $C2[TIE]$ ,  $C2[TCIE]$ , and  $C2[RIE]$  as desired.



At this time, the UART will start listening for an initial character. After being identified, it will automatically adjust S2[MSBF], C3[TXINV], and S2[RXINV]. The software must then receive and process an answer to reset. Upon processing the answer to reset, the software must write to set C2[RE] = 0 and C2[TE] = 0. The software should then adjust 7816 specific and UART generic parameters to match and configure data that was received during the answer on reset period. After the new settings have been programmed, including the new baud rate and C7816[TTYTYPE], C2[RE] and C2[TE] can be reenabled as required.

### 36.9.1.1 Transmission procedure for (C7816[TTYTYPE] = 0)

When the protocol selected is C7816[TTYTYPE] = 0, it is assumed that the software has a prior knowledge of who should be transmitting and receiving. Therefore, no mechanism is provided for automated transmission/receipt control. The software must monitor S1[TDRE], or configure for an interrupt, and provide additional data for transmission, as appropriate. Additionally, software should set C2[TE] = 1 and control TXDIR whenever it is the UART's turn to transmit information. For ease of monitoring, it is suggested that only data be transmitted until the next receiver/transmit switchover is loaded into the transmit buffer.

### 36.9.1.2 Transmission procedure for (C7816[TTYTYPE] = 1)

When the protocol selected is C7816[TTYTYPE] = 1, data is transferred in blocks. Before starting a transmission, the software must write the size, in number of bytes, for the Information Field portion of the block into TLEN. If a CRC is being transmitted for the block, the value in TLEN must be one more than the size of the information field. The software must then set C2[TE] = 1 and C2[RE] = 1. The software must then monitor S1[TDRE]/interrupt and write the prologue, information, and epilogue field to the transmit buffer. TLEN automatically decrements, except for prologue bytes and the final epilogue byte. When the final epilogue byte has been transmitted, the UART automatically clears C2[TE] and C3[TXDIR] to 0, and the UART automatically starts capturing the response to the block that was transmitted. After the software has detected the receipt of the response, the transmission process must be repeated as needed with sufficient urgency to ensure that the block wait time and character wait times are not violated.

## 36.9.2 Initialization sequence (non ISO-7816)

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. 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, 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.

### 36.9.3 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.

#### 36.9.3.1 Overrun operation

The assertion of S1[OR] indicates that a significant event has occurred. The assertion indicates that received data has been lost because there was a lack of room to store it in the data buffer. Therefore, while S1[OR] is set, no further data is stored in the data buffer until S1[OR] is cleared. This ensures that the application will be able to handle the overrun condition.

In most applications, because the total amount of lost data is known, the application will attempt to return the system to a known state. Before S1[OR] is cleared, all received data will be dropped. For this, the software does the following.

1. Remove data from the receive data buffer. This could be done by reading data from the data buffer and processing it.
2. Clear S1[OR].

Note that, in some applications, if an overrun event is responded to fast enough, the lost data can be recovered. For example, when C7816[ISO\_7816E] is asserted, C7816[TTYTYPE]=1 and C7816[ONACK] = 1, the application may reasonably be able to determine whether the lost data will be resent by the device. In this scenario, flushing the receiver data buffer may not be required. Rather, if S1[OR] is cleared, the lost data may be resent and therefore may be recoverable.

### 36.9.4 Overrun NACK considerations

When C7816[ISO\_7816E] is enabled and C7816[TTYTYPE] = 0, the retransmission feature of the 7816 protocol can be used to help avoid lost data when the data buffer overflows. Using C7816[ONACK], the module can be programmed to issue a NACK on an overflow event. Assuming that the smartcard device has implemented retransmission, the lost data will be retransmitted. While useful, there is a programming implication that may require special consideration. The need to transmit a NACK must be determined and committed to prior to the dataword being fully received. While the NACK is being

received, it is possible that the application code will read the data buffer such that sufficient room will be made to store the dataword that is being NACKed. Even if room has been made in the data buffer after the transmission of a NACK is completed, the received data will always be discarded as a result of an overflow and the ET7816[RXTHRESHOLD] value will be incremented by one. However, if sufficient space now exists to write the received data which was NACK'ed, S1[OR] will be blocked and kept from asserting.

### 36.9.5 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.

### 36.9.6 Clearing 7816 wait timer (WT, BWT, CWT) interrupts

The 7816 wait timer interrupts associated with IS7816[WT], IS7816[BWT], and IS7816[CWT] will automatically reassert if they are cleared and the wait time is still violated. This behavior is similar to most of the other interrupts on the UART. In most cases, if the condition that caused the interrupt to trigger still exists when the interrupt is cleared, then the interrupt will reassert. For example, consider the following scenario:

1. IS7816[WT] is programmed to assert after 9600 cycles of unresponsiveness.
2. The 9600 cycles pass without a response resulting in the WT interrupt asserting.
3. The IS7816[WT] is cleared at cycle 9700 by the interrupt service routine.
4. After the WT interrupt has been cleared, the smartcard remains unresponsive. At cycle 9701 the WT interrupt will be reasserted.

If the intent of clearing the interrupt is such that it does not reassert, the interrupt service routine must remove or clear the condition that originally caused the interrupt to assert prior to clearing the interrupt. There are multiple ways that this can be accomplished, including ensuring that an event that results in the wait timer resetting occurs, such as, the transmission of another packet.

### 36.9.7 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. 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 37

## FlexIO

### 37.1 Chip-specific FlexIO information

#### 37.1.1 FlexIO

This section summarize the features and module configurations of FlexIO

#### 37.1.2 Clock options

The FlexIO blocks are clocked from a single FlexIO clock that can be selected from OSCERCLK, MCGIRCLK, or MCGPCLK (IRC48M clock) . The selected source is controlled by SIM\_SOPT2[FLEXIOSRC] .

#### 37.1.3 Trigger options

FlexIO has a selectable trigger input source controlled by FlexIO\_TIMCTLn[TRGSEL] (4-bit field) to use for starting the counter and/or reloading the counter. The options available are shown in the following table.

**Table 37-1. FlexIO trigger options**

FlexIO_TIMCTLn[TRGSEL] (4-bit field)	Selected source
0000	External trigger pin input (EXTRG_IN)
0001	CMP0 output
0010	Reserved
0011	Reserved
0100	PIT trigger 0
0101	PIT trigger 1
0110	Reserved

*Table continues on the next page...*

**Table 37-1. FlexIO trigger options (continued)**

FlexIO_TIMCTLn[TRGSEL] (4-bit field)	Selected source
0111	Reserved
1000	TPM0 overflow
1001	TPM1 overflow
1010	TPM2 overflow
1011	Reserved
1100	RTC alarm
1101	RTC seconds
1110	LPTMR trigger
1111	Reserved

## 37.2 Introduction

### 37.2.1 Overview

The FlexIO is a highly configurable module providing a wide range of functionality including:

- Emulation of a variety of serial communication protocols
- Flexible 16-bit timers with support for a variety of trigger, reset, enable and disable conditions

These functions are provided by the FlexIO while adhering to the following key objectives:

- Low software/CPU overhead: less overhead than software bit-banging, more overhead than dedicated peripheral IP.
- Area/Power efficient implementation: more efficient than integrating multiple peripherals for each desired protocol.

### 37.2.2 Features

The FlexIO module is capable of supporting a wide range of protocols including, but not limited to:

- UART
- I2C
- SPI



- I2S
- PWM/Waveform generation

The following key features are provided:

- Array of 32-bit shift registers with transmit, receive and data match modes
- Double buffered shifter operation for continuous data transfer
- Shifter concatenation to support large transfer sizes
- Automatic start/stop bit generation
- Interrupt, DMA or polled transmit/receive operation
- Programmable baud rates independent of bus clock frequency, with support for asynchronous operation during stop modes
- Highly flexible 16-bit timers with support for a variety of internal or external trigger, reset, enable and disable conditions

### 37.2.3 Block Diagram

The following diagram gives a high-level overview of the configuration of FlexIO timers and shifters.

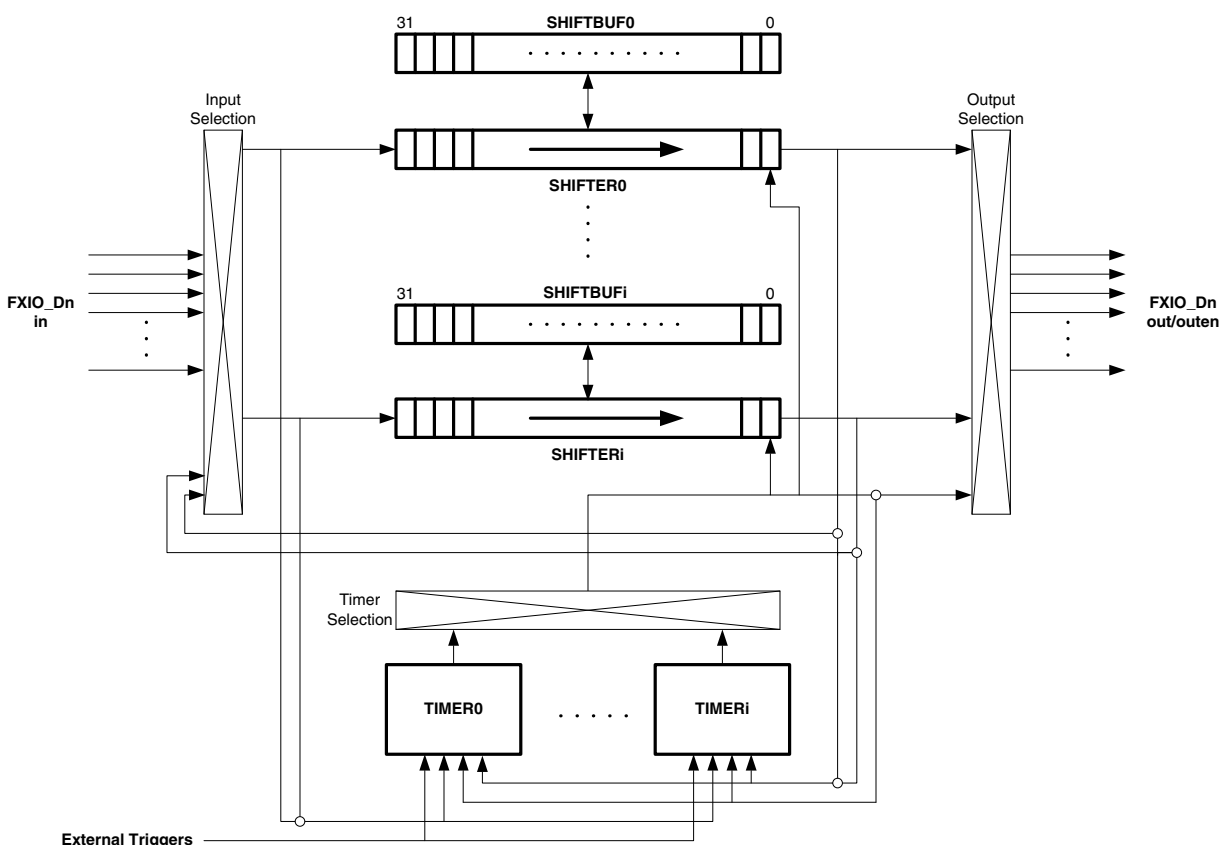


Figure 37-1. FlexIO block diagram

## 37.2.4 Modes of operation

The FlexIO module supports the chip modes described in the following table.

**Table 37-2. Chip modes supported by the FlexIO module**

Chip mode	FlexIO Operation
Run	Normal operation
Stop/Wait	Can continue operating provided the Doze Enable bit (CTRL[DOZEN]) is set and the FlexIO is using an external or internal clock source which remains operating during stop/wait modes.
Low Leakage Stop	The Doze Enable (CTRL[DOZEN]) bit is ignored and the FlexIO will wait for all Timers to complete any pending operation before acknowledging low leakage mode entry.
Debug	Can continue operating provided the Debug Enable bit (CTRL[DBGGE]) is set.

## 37.2.5 FlexIO Signal Descriptions

Signal	Description	I/O
FXIO_Dn (n=0...7)	Bidirectional FlexIO Shifter and Timer pin inputs/outputs	I/O

## 37.3 Memory Map and Registers

**FLEXIO memory map**

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4005_F000	Version ID Register (FLEXIO_VERID)	32	R	0100_0000h	<a href="#">37.3.1/700</a>
4005_F004	Parameter Register (FLEXIO_PARAM)	32	R	<a href="#">See section</a>	<a href="#">37.3.2/701</a>
4005_F008	FlexIO Control Register (FLEXIO_CTRL)	32	R/W	0000_0000h	<a href="#">37.3.3/702</a>
4005_F010	Shifter Status Register (FLEXIO_SHIFTSTAT)	32	w1c	0000_0000h	<a href="#">37.3.4/703</a>
4005_F014	Shifter Error Register (FLEXIO_SHIFTEERR)	32	w1c	0000_0000h	<a href="#">37.3.5/704</a>
4005_F018	Timer Status Register (FLEXIO_TIMSTAT)	32	w1c	0000_0000h	<a href="#">37.3.6/704</a>
4005_F020	Shifter Status Interrupt Enable (FLEXIO_SHIFTSIEN)	32	R/W	0000_0000h	<a href="#">37.3.7/705</a>
4005_F024	Shifter Error Interrupt Enable (FLEXIO_SHIFTEIEN)	32	R/W	0000_0000h	<a href="#">37.3.8/706</a>
4005_F028	Timer Interrupt Enable Register (FLEXIO_TIMIEN)	32	R/W	0000_0000h	<a href="#">37.3.9/706</a>

*Table continues on the next page...*

## FLEXIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4005_F030	Shifter Status DMA Enable (FLEXIO_SHIFTSDEN)	32	R/W	0000_0000h	37.3.10/ 707
4005_F080	Shifter Control N Register (FLEXIO_SHIFTCTL0)	32	R/W	0000_0000h	37.3.11/ 707
4005_F084	Shifter Control N Register (FLEXIO_SHIFTCTL1)	32	R/W	0000_0000h	37.3.11/ 707
4005_F088	Shifter Control N Register (FLEXIO_SHIFTCTL2)	32	R/W	0000_0000h	37.3.11/ 707
4005_F08C	Shifter Control N Register (FLEXIO_SHIFTCTL3)	32	R/W	0000_0000h	37.3.11/ 707
4005_F100	Shifter Configuration N Register (FLEXIO_SHIFTCFG0)	32	R/W	0000_0000h	37.3.12/ 709
4005_F104	Shifter Configuration N Register (FLEXIO_SHIFTCFG1)	32	R/W	0000_0000h	37.3.12/ 709
4005_F108	Shifter Configuration N Register (FLEXIO_SHIFTCFG2)	32	R/W	0000_0000h	37.3.12/ 709
4005_F10C	Shifter Configuration N Register (FLEXIO_SHIFTCFG3)	32	R/W	0000_0000h	37.3.12/ 709
4005_F200	Shifter Buffer N Register (FLEXIO_SHIFTBUF0)	32	R/W	0000_0000h	37.3.13/ 710
4005_F204	Shifter Buffer N Register (FLEXIO_SHIFTBUF1)	32	R/W	0000_0000h	37.3.13/ 710
4005_F208	Shifter Buffer N Register (FLEXIO_SHIFTBUF2)	32	R/W	0000_0000h	37.3.13/ 710
4005_F20C	Shifter Buffer N Register (FLEXIO_SHIFTBUF3)	32	R/W	0000_0000h	37.3.13/ 710
4005_F280	Shifter Buffer N Bit Swapped Register (FLEXIO_SHIFTBUFBIS0)	32	R/W	0000_0000h	37.3.14/ 711
4005_F284	Shifter Buffer N Bit Swapped Register (FLEXIO_SHIFTBUFBIS1)	32	R/W	0000_0000h	37.3.14/ 711
4005_F288	Shifter Buffer N Bit Swapped Register (FLEXIO_SHIFTBUFBIS2)	32	R/W	0000_0000h	37.3.14/ 711
4005_F28C	Shifter Buffer N Bit Swapped Register (FLEXIO_SHIFTBUFBIS3)	32	R/W	0000_0000h	37.3.14/ 711
4005_F300	Shifter Buffer N Byte Swapped Register (FLEXIO_SHIFTBUFBYS0)	32	R/W	0000_0000h	37.3.15/ 711
4005_F304	Shifter Buffer N Byte Swapped Register (FLEXIO_SHIFTBUFBYS1)	32	R/W	0000_0000h	37.3.15/ 711
4005_F308	Shifter Buffer N Byte Swapped Register (FLEXIO_SHIFTBUFBYS2)	32	R/W	0000_0000h	37.3.15/ 711
4005_F30C	Shifter Buffer N Byte Swapped Register (FLEXIO_SHIFTBUFBYS3)	32	R/W	0000_0000h	37.3.15/ 711
4005_F380	Shifter Buffer N Bit Byte Swapped Register (FLEXIO_SHIFTBUFBBS0)	32	R/W	0000_0000h	37.3.16/ 712

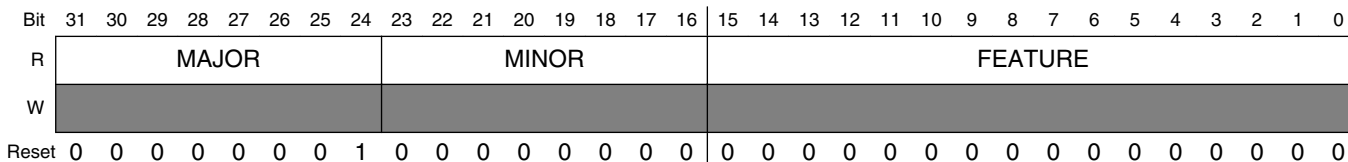
Table continues on the next page...

**FLEXIO memory map (continued)**

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4005_F384	Shifter Buffer N Bit Byte Swapped Register (FLEXIO_SHIFTBUFBBS1)	32	R/W	0000_0000h	37.3.16/712
4005_F388	Shifter Buffer N Bit Byte Swapped Register (FLEXIO_SHIFTBUFBBS2)	32	R/W	0000_0000h	37.3.16/712
4005_F38C	Shifter Buffer N Bit Byte Swapped Register (FLEXIO_SHIFTBUFBBS3)	32	R/W	0000_0000h	37.3.16/712
4005_F400	Timer Control N Register (FLEXIO_TIMCTL0)	32	R/W	0000_0000h	37.3.17/712
4005_F404	Timer Control N Register (FLEXIO_TIMCTL1)	32	R/W	0000_0000h	37.3.17/712
4005_F408	Timer Control N Register (FLEXIO_TIMCTL2)	32	R/W	0000_0000h	37.3.17/712
4005_F40C	Timer Control N Register (FLEXIO_TIMCTL3)	32	R/W	0000_0000h	37.3.17/712
4005_F480	Timer Configuration N Register (FLEXIO_TIMCFG0)	32	R/W	0000_0000h	37.3.18/714
4005_F484	Timer Configuration N Register (FLEXIO_TIMCFG1)	32	R/W	0000_0000h	37.3.18/714
4005_F488	Timer Configuration N Register (FLEXIO_TIMCFG2)	32	R/W	0000_0000h	37.3.18/714
4005_F48C	Timer Configuration N Register (FLEXIO_TIMCFG3)	32	R/W	0000_0000h	37.3.18/714
4005_F500	Timer Compare N Register (FLEXIO_TIMCMP0)	32	R/W	0000_0000h	37.3.19/716
4005_F504	Timer Compare N Register (FLEXIO_TIMCMP1)	32	R/W	0000_0000h	37.3.19/716
4005_F508	Timer Compare N Register (FLEXIO_TIMCMP2)	32	R/W	0000_0000h	37.3.19/716
4005_F50C	Timer Compare N Register (FLEXIO_TIMCMP3)	32	R/W	0000_0000h	37.3.19/716

**37.3.1 Version ID Register (FLEXIO\_VERID)**

Address: 4005\_F000h base + 0h offset = 4005\_F000h



### FLEXIO\_VERID field descriptions

Field	Description
31–24 MAJOR	Major Version Number This read only field returns the major version number for the module specification.
23–16 MINOR	Minor Version Number This read only field returns the minor version number for the module specification.
FEATURE	Feature Specification Number This read only field returns the feature set number.  0x0000 Standard features implemented. 0x0001 Supports state, logic and parallel modes.

### 37.3.2 Parameter Register (FLEXIO\_PARAM)

Address: 4005\_F000h base + 4h offset = 4005\_F004h

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	TRIGGER								PIN								TIMER								SHIFTER															
W	[Shaded]																																							
Reset	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

### FLEXIO\_PARAM field descriptions

Field	Description
31–24 TRIGGER	Trigger Number Number of external triggers implemented.
23–16 PIN	Pin Number Number of Pins implemented.
15–8 TIMER	Timer Number Number of Timers implemented.
SHIFTER	Shifter Number Number of Shifters implemented.

### 37.3.3 FlexIO Control Register (FLEXIO\_CTRL)

Address: 4005\_F000h base + 8h offset = 4005\_F008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	DOZEN		DBGE		0												
W	DOZEN		DBGE		[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														FASTACC	SWRST	FLEXEN
W	[Reserved]														FASTACC	SWRST	FLEXEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

#### FLEXIO\_CTRL field descriptions

Field	Description
31 DOZEN	<p>Doze Enable</p> <p>Disables FlexIO operation in Doze modes. This field is ignored and the FlexIO always disabled in low-leakage stop modes.</p> <p>0 FlexIO enabled in Doze modes. 1 FlexIO disabled in Doze modes.</p>
30 DBGE	<p>Debug Enable</p> <p>Enables FlexIO operation in Debug mode.</p> <p>0 FlexIO is disabled in debug modes. 1 FlexIO is enabled in debug modes</p>
29–3 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
2 FASTACC	<p>Fast Access</p> <p>Enables fast register accesses to FlexIO registers, but requires the FlexIO clock to be at least twice the frequency of the bus clock.</p> <p>0 Configures for normal register accesses to FlexIO 1 Configures for fast register accesses to FlexIO</p>
1 SWRST	<p>Software Reset</p> <p>The FlexIO Control Register is not affected by the software reset, all other logic in the FlexIO is affected by the software reset and register accesses are ignored until this bit is cleared. This register bit will remain set until cleared by software, and the reset has cleared in the FlexIO clock domain.</p>

Table continues on the next page...

### FLEXIO\_CTRL field descriptions (continued)

Field	Description
	0 Software reset is disabled 1 Software reset is enabled, all FlexIO registers except the Control Register are reset.
0 FLEXEN	FlexIO Enable 0 FlexIO module is disabled. 1 FlexIO module is enabled.

### 37.3.4 Shifter Status Register (FLEXIO\_SHIFTSTAT)

Address: 4005\_F000h base + 10h offset = 4005\_F010h

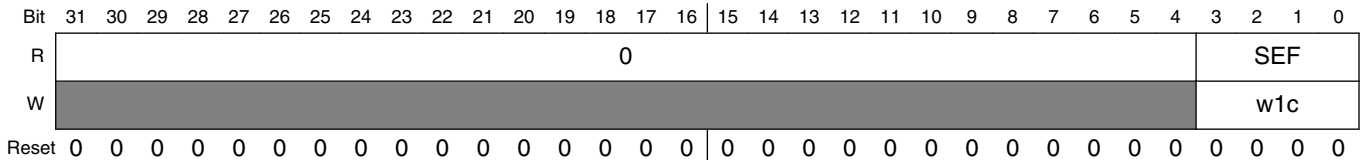
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																SSF															
W																			w1c													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

### FLEXIO\_SHIFTSTAT field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSF	<p>Shifter Status Flag</p> <p>The shifter status flag is updated when one of the following events occurs:</p> <p>For SMOD=Receive, the status flag is set when SHIFTBUF has been loaded with data from Shifter (SHIFTBUF is full), and the status flag is cleared when SHIFTBUF register is read.</p> <p>For SMOD=Transmit, the status flag is set when SHIFTBUF data has been transferred to the Shifter (SHIFTBUF is empty) or when initially configured for SMOD=Transmit, and the status flag is cleared when the SHIFTBUF register is written.</p> <p>For SMOD=Match Store, the status flag is set when a match has occurred between SHIFTBUF and Shifter, and the status flag is cleared when the SHIFTBUF register is read.</p> <p>For SMOD=Match Continuous, returns the current match result between the SHIFTBUF and Shifter.</p> <p>The status flag can also be cleared by writing a logic one to the flag for all modes except Match Continuous.</p> <p>0 Status flag is clear 1 Status flag is set</p>

### 37.3.5 Shifter Error Register (FLEXIO\_SHIFTErr)

Address: 4005\_F000h base + 14h offset = 4005\_F014h

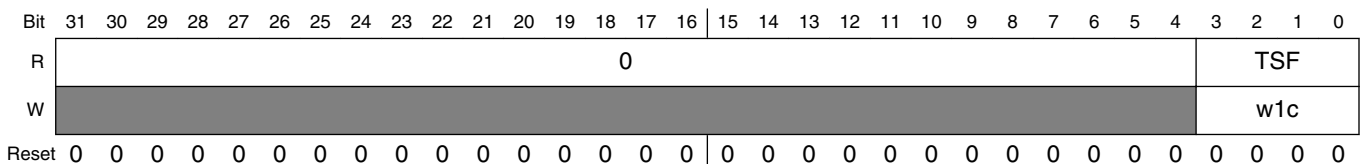


#### FLEXIO\_SHIFTErr field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEF	<p>Shifter Error Flags</p> <p>The shifter error flag is set when one of the following events occurs:</p> <p>For SMOD=Receive, indicates Shifter was ready to store new data into SHIFTBUF before the previous data was read from SHIFTBUF (SHIFTBUF Overrun), or indicates that the received start or stop bit does not match the expected value.</p> <p>For SMOD=Transmit, indicates Shifter was ready to load new data from SHIFTBUF before new data had been written into SHIFTBUF (SHIFTBUF Underrun).</p> <p>For SMOD=Match Store, indicates a match event occurred before the previous match data was read from SHIFTBUF (SHIFTBUF Overrun).</p> <p>For SMOD=Match Continuous, the error flag is set when a match has occurred between SHIFTBUF and Shifter.</p> <p>Can be cleared by writing logic one to the flag. For SMOD=Match Continuous, can also be cleared when the SHIFTBUF register is read.</p> <p>0 Shifter Error Flag is clear 1 Shifter Error Flag is set</p>

### 37.3.6 Timer Status Register (FLEXIO\_TIMSTAT)

Address: 4005\_F000h base + 18h offset = 4005\_F018h





### FLEXIO\_TIMSTAT field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TSF	<p>Timer Status Flags</p> <p>The timer status flag sets depending on the timer mode, and can be cleared by writing logic one to the flag.</p> <p>In 8-bit counter mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements, this also causes the counter to reload with the value in the compare register.</p> <p>In 8-bit PWM mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements, this also causes the counter to reload with the value in the compare register..</p> <p>In 16-bit counter mode, the timer status flag is set when the 16-bit counter equals zero and decrements, this also causes the counter to reload with the value in the compare register..</p> <p>0 Timer Status Flag is clear 1 Timer Status Flag is set</p>

### 37.3.7 Shifter Status Interrupt Enable (FLEXIO\_SHIFTSIEN)

Address: 4005\_F000h base + 20h offset = 4005\_F020h

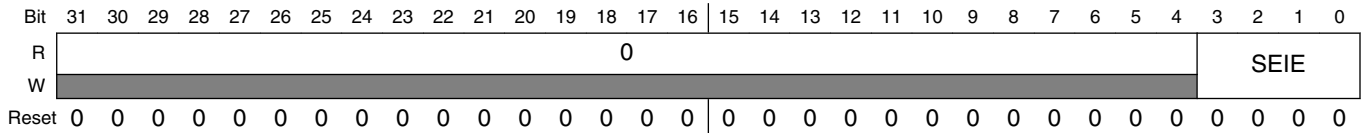
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																SSIE															
W	0																0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

### FLEXIO\_SHIFTSIEN field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSIE	<p>Shifter Status Interrupt Enable</p> <p>Enables interrupt generation when corresponding SSF is set.</p> <p>0 Shifter Status Flag interrupt disabled 1 Shifter Status Flag interrupt enabled</p>

### 37.3.8 Shifter Error Interrupt Enable (FLEXIO\_SHIFTEIEN)

Address: 4005\_F000h base + 24h offset = 4005\_F024h

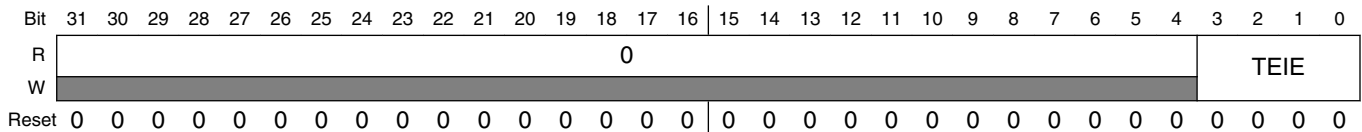


#### FLEXIO\_SHIFTEIEN field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEIE	Shifter Error Interrupt Enable  Enables interrupt generation when corresponding SEF is set.  0 Shifter Error Flag interrupt disabled 1 Shifter Error Flag interrupt enabled

### 37.3.9 Timer Interrupt Enable Register (FLEXIO\_TIMIEN)

Address: 4005\_F000h base + 28h offset = 4005\_F028h



#### FLEXIO\_TIMIEN field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TEIE	Timer Status Interrupt Enable  Enables interrupt generation when corresponding TSF is set.  0 Timer Status Flag interrupt is disabled 1 Timer Status Flag interrupt is enabled

### 37.3.10 Shifter Status DMA Enable (FLEXIO\_SHIFTSDEN)

Address: 4005\_F000h base + 30h offset = 4005\_F030h

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																SSDE															
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

#### FLEXIO\_SHIFTSDEN field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSDE	Shifter Status DMA Enable  Enables DMA request generation when corresponding SSF is set.  0 Shifter Status Flag DMA request is disabled 1 Shifter Status Flag DMA request is enabled

### 37.3.11 Shifter Control N Register (FLEXIO\_SHIFTCTL<sub>n</sub>)

Address: 4005\_F000h base + 80h offset + (4d × 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								TIMPOL	0								
W	0									TIMSEL	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								PINPOL	0								
W	0									PINSEL	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

FLEXIO\_SHIFTCTL $n$  field descriptions

Field	Description
31–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 TIMSEL	Timer Select Selects which Timer is used for controlling the logic/shift register and generating the Shift clock.
23 TIMPOL	Timer Polarity 0 Shift on posedge of Shift clock 1 Shift on negedge of Shift clock
22–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–16 PINCFG	Shifter Pin Configuration 00 Shifter pin output disabled 01 Shifter pin open drain or bidirectional output enable 10 Shifter pin bidirectional output data 11 Shifter pin output
15–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 PINSEL	Shifter Pin Select Selects which pin is used by the Shifter input or output.
7 PINPOL	Shifter Pin Polarity 0 Pin is active high 1 Pin is active low
6–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SMOD	Shifter Mode Configures the mode of the Shifter. 000 Disabled. 001 Receive mode. Captures the current Shifter content into the SHIFTBUF on expiration of the Timer. 010 Transmit mode. Load SHIFTBUF contents into the Shifter on expiration of the Timer. 011 Reserved. 100 Match Store mode. Shifter data is compared to SHIFTBUF content on expiration of the Timer. 101 Match Continuous mode. Shifter data is continuously compared to SHIFTBUF contents. 110 Reserved. 111 Reserved.

### 37.3.12 Shifter Configuration N Register (FLEXIO\_SHIFTCFGn)

Address: 4005\_F000h base + 100h offset + (4d × 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								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	0	SSTOP		0		SSTART	
W	[Reserved]								INSRC	[Reserved]	SSTOP		[Reserved]	SSTART		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

#### FLEXIO\_SHIFTCFGn field descriptions

Field	Description
31–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 INSRC	Input Source Selects the input source for the shifter.  0 Pin 1 Shifter N+1 Output
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.
5–4 SSTOP	Shifter Stop bit  For SMOD=Transmit, this field allows automatic stop bit insertion if the selected timer has also enabled a stop bit.  For SMOD=Receive or Match Store, this field allows automatic stop bit checking if the selected timer has also enabled a stop bit.  00 Stop bit disabled for transmitter/receiver/match store 01 Reserved for transmitter/receiver/match store 10 Transmitter outputs stop bit value 0 on store, receiver/match store sets error flag if stop bit is not 0 11 Transmitter outputs stop bit value 1 on store, receiver/match store sets error flag if stop bit is not 1

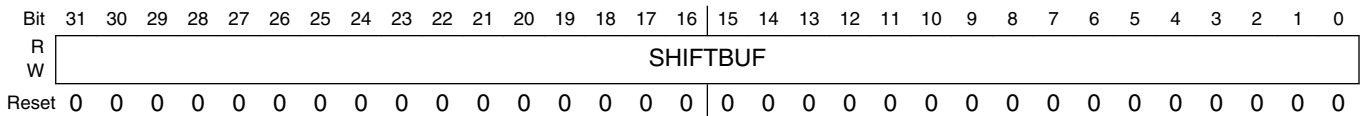
Table continues on the next page...

**FLEXIO\_SHIFTCFGn field descriptions (continued)**

Field	Description
3-2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SSTART	<p>Shifter Start bit</p> <p>For SMOD=Transmit, this field allows automatic start bit insertion if the selected timer has also enabled a start bit.</p> <p>For SMOD=Receive or Match Store, this field allows automatic start bit checking if the selected timer has also enabled a start bit.</p> <p>00 Start bit disabled for transmitter/receiver/match store, transmitter loads data on enable                      01 Start bit disabled for transmitter/receiver/match store, transmitter loads data on first shift                      10 Transmitter outputs start bit value 0 before loading data on first shift, receiver/match store sets error flag if start bit is not 0                      11 Transmitter outputs start bit value 1 before loading data on first shift, receiver/match store sets error flag if start bit is not 1</p>

**37.3.13 Shifter Buffer N Register (FLEXIO\_SHIFTBUFn)**

Address: 4005\_F000h base + 200h offset + (4d × i), where i=0d to 3d

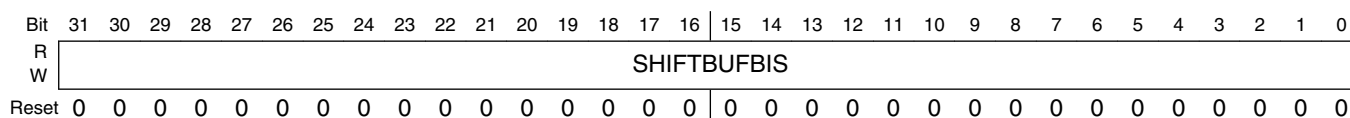


**FLEXIO\_SHIFTBUFn field descriptions**

Field	Description
SHIFTBUF	<p>Shift Buffer</p> <p>Shift buffer data is used for a variety of functions depending on the SMOD setting:</p> <p>For SMOD=Receive, Shifter data is transferred into SHIFTBUF at the expiration of Timer.</p> <p>For SMOD=Transmit, SHIFTBUF data is transferred into the Shifter before the Timer begins.</p> <p>For SMOD=Match Store/Continuous, SHIFTBUF[31:16] contains the data to be matched with the Shifter contents. The Match is checked either continuously (Match Continuous mode) or when the Timer expires (Match Store mode). SHIFTBUF[15:0] can be used to mask the match result (1=mask, 0=no mask). In Match Store mode, Shifter data [31:16] is written to SHIFTBUF[31:16] whenever a match event occurs.</p>

### 37.3.14 Shifter Buffer N Bit Swapped Register (FLEXIO\_SHIFTBUFBIS<sub>n</sub>)

Address: 4005\_F000h base + 280h offset + (4d × i), where i=0d to 3d

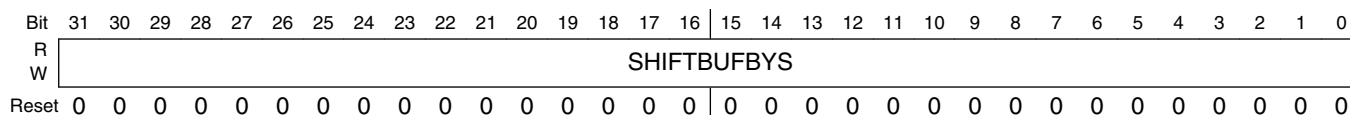


#### FLEXIO\_SHIFTBUFBIS<sub>n</sub> field descriptions

Field	Description
SHIFTBUFBIS	Shift Buffer  Alias to SHIFTBUF register, except reads/writes to this register are bit swapped. Reads return SHIFTBUF[0:31].

### 37.3.15 Shifter Buffer N Byte Swapped Register (FLEXIO\_SHIFTBUFBYS<sub>n</sub>)

Address: 4005\_F000h base + 300h offset + (4d × i), where i=0d to 3d

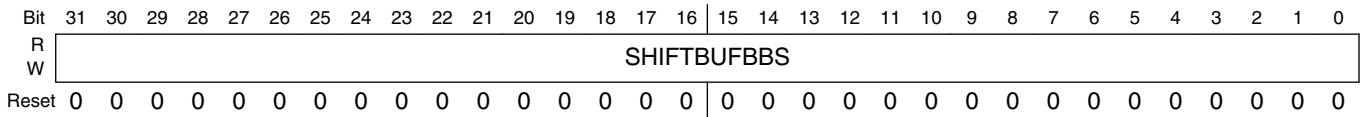


#### FLEXIO\_SHIFTBUFBYS<sub>n</sub> field descriptions

Field	Description
SHIFTBUFBYS	Shift Buffer  Alias to SHIFTBUF register, except reads/writes to this register are byte swapped. Reads return { SHIFTBUF[7:0], SHIFTBUF[15:8], SHIFTBUF[23:16], SHIFTBUF[31:24] }.

### 37.3.16 Shifter Buffer N Bit Byte Swapped Register (FLEXIO\_SHIFTBUFBBS<sub>n</sub>)

Address: 4005\_F000h base + 380h offset + (4d × i), where i=0d to 3d

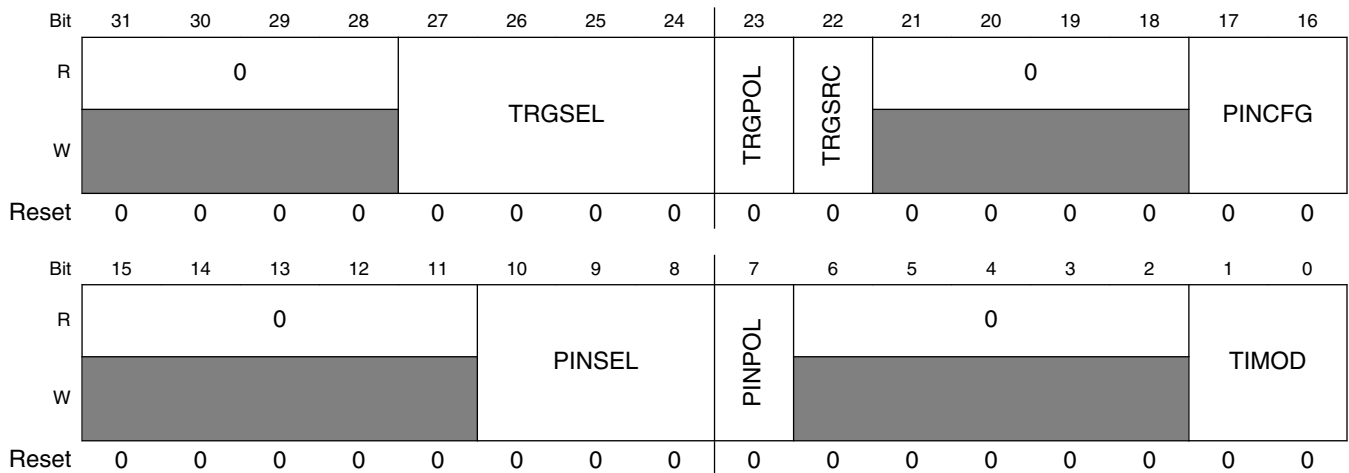


**FLEXIO\_SHIFTBUFBBS<sub>n</sub> field descriptions**

Field	Description
SHIFTBUFBBS	Shift Buffer  Alias to SHIFTBUF register, except reads/writes to this register are bit swapped within each byte. Reads return { SHIFTBUF[24:31], SHIFTBUF[16:23], SHIFTBUF[8:15], SHIFTBUF[0:7] }.

### 37.3.17 Timer Control N Register (FLEXIO\_TIMCTL<sub>n</sub>)

Address: 4005\_F000h base + 400h offset + (4d × i), where i=0d to 3d



**FLEXIO\_TIMCTL<sub>n</sub> 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 TRGSEL	Trigger Select

*Table continues on the next page...*



FLEXIO\_TIMCTL<sub>n</sub> field descriptions (continued)

Field	Description
	<p>The valid values for TRGSEL will depend on the FLEXIO_PARAM register. When TRGSRC = 1, the valid values for N will depend on PIN, TIMER, SHIFTER fields in the FLEXIO_PARAM register. When TRGSRC = 0, valid values for N will depend on TRIGGER field in FLEXIO_PARAM register. Refer to the chip configuration section for external trigger selection. The internal trigger selection is configured as follows.</p> <p>{N,00} pin 2N input            {N,01} shifter N status flag            {N,10} pin 2N+1 input            {N,11} timer N trigger output</p>
23 TRGPOL	<p>Trigger Polarity</p> <p>0 Trigger active high            1 Trigger active low</p>
22 TRGSRC	<p>Trigger Source</p> <p>0 External trigger selected            1 Internal trigger selected</p>
21–18 Reserved	<p>This field is reserved.            This read-only field is reserved and always has the value 0.</p>
17–16 PINCFG	<p>Timer Pin Configuration</p> <p>00 Timer pin output disabled            01 Timer pin open drain or bidirectional output enable            10 Timer pin bidirectional output data            11 Timer pin output</p>
15–11 Reserved	<p>This field is reserved.            This read-only field is reserved and always has the value 0.</p>
10–8 PINSEL	<p>Timer Pin Select</p> <p>Selects which pin is used by the Timer input or output.</p>
7 PINPOL	<p>Timer Pin Polarity</p> <p>0 Pin is active high            1 Pin is active low</p>
6–2 Reserved	<p>This field is reserved.            This read-only field is reserved and always has the value 0.</p>
TIMOD	<p>Timer Mode</p> <p>In 8-bit counter mode, the lower 8-bits of the counter and compare register are used to configure the baud rate of the timer shift clock and the upper 8-bits are used to configure the shifter bit count.</p> <p>In 8-bit PWM mode, the lower 8-bits of the counter and compare register are used to configure the high period of the timer shift clock and the upper 8-bits are used to configure the low period of the timer shift clock. The shifter bit count is configured using another timer or external signal.</p> <p>In 16-bit counter mode, the full 16-bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count.</p> <p>00 Timer Disabled.            01 Dual 8-bit counters baud/bit mode.</p>

*Table continues on the next page...*

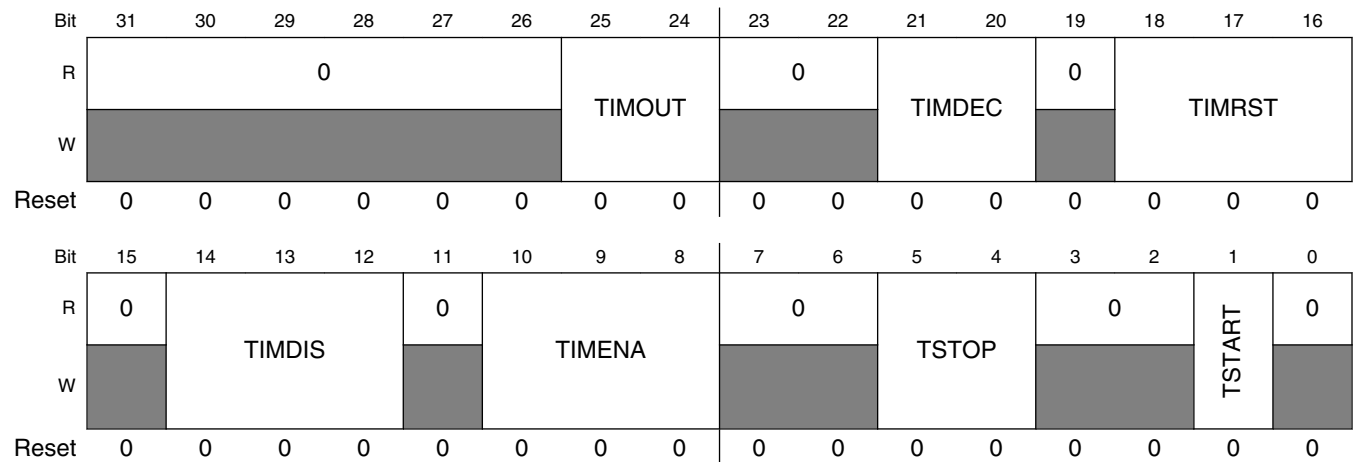
**FLEXIO\_TIMCTLn field descriptions (continued)**

Field	Description
10	Dual 8-bit counters PWM mode.
11	Single 16-bit counter mode.

**37.3.18 Timer Configuration N Register (FLEXIO\_TIMCFGn)**

The options to enable or disable the timer using the Timer N-1 enable or disable are reserved when N is evenly divisible by 4 (eg: Timer 0).

Address: 4005\_F000h base + 480h offset + (4d × i), where i=0d to 3d



**FLEXIO\_TIMCFGn field descriptions**

Field	Description
31–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 TIMOUT	Timer Output Configures the initial state of the Timer Output and whether it is affected by the Timer reset.  00 Timer output is logic one when enabled and is not affected by timer reset 01 Timer output is logic zero when enabled and is not affected by timer reset 10 Timer output is logic one when enabled and on timer reset 11 Timer output is logic zero when enabled and on timer reset
23–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–20 TIMDEC	Timer Decrement Configures the source of the Timer decrement and the source of the Shift clock.  00 Decrement counter on FlexIO clock, Shift clock equals Timer output. 01 Decrement counter on Trigger input (both edges), Shift clock equals Timer output.

Table continues on the next page...

## FLEXIO\_TIMCFGn field descriptions (continued)

Field	Description
	10 Decrement counter on Pin input (both edges), Shift clock equals Pin input. 11 Decrement counter on Trigger input (both edges), Shift clock equals Trigger input.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–16 TIMRST	Timer Reset  Configures the condition that causes the timer counter (and optionally the timer output) to be reset. In 8-bit counter mode, the timer reset will only reset the lower 8-bits that configure the baud rate. In all other modes, the timer reset will reset the full 16-bits of the counter.  000 Timer never reset 001 Reserved 010 Timer reset on Timer Pin equal to Timer Output 011 Timer reset on Timer Trigger equal to Timer Output 100 Timer reset on Timer Pin rising edge 101 Reserved 110 Timer reset on Trigger rising edge 111 Timer reset on Trigger rising or falling edge
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–12 TIMDIS	Timer Disable  Configures the condition that causes the Timer to be disabled and stop decrementing.  000 Timer never disabled 001 Timer disabled on Timer N-1 disable 010 Timer disabled on Timer compare 011 Timer disabled on Timer compare and Trigger Low 100 Timer disabled on Pin rising or falling edge 101 Timer disabled on Pin rising or falling edge provided Trigger is high 110 Timer disabled on Trigger falling edge 111 Reserved
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 TIMENA	Timer Enable  Configures the condition that causes the Timer to be enabled and start decrementing.  000 Timer always enabled 001 Timer enabled on Timer N-1 enable 010 Timer enabled on Trigger high 011 Timer enabled on Trigger high and Pin high 100 Timer enabled on Pin rising edge 101 Timer enabled on Pin rising edge and Trigger high 110 Timer enabled on Trigger rising edge 111 Timer enabled on Trigger rising or falling edge
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

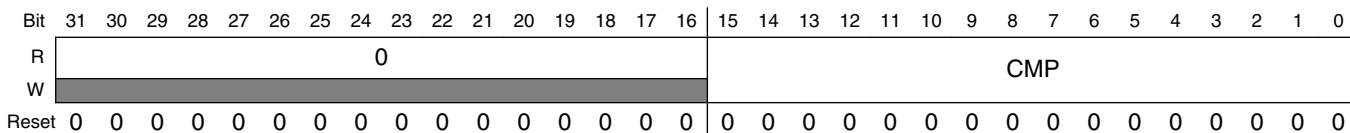
*Table continues on the next page...*

**FLEXIO\_TIMCFGn field descriptions (continued)**

Field	Description
5-4 TSTOP	<p>Timer Stop Bit</p> <p>The stop bit can be added on a timer compare (between each word) or on a timer disable. When stop bit is enabled, configured shifters will output the contents of the stop bit when the timer is disabled. When stop bit is enabled on timer disable, the timer remains disabled until the next rising edge of the shift clock. If configured for both timer compare and timer disable, only one stop bit is inserted on timer disable.</p> <p>00 Stop bit disabled                      01 Stop bit is enabled on timer compare                      10 Stop bit is enabled on timer disable                      11 Stop bit is enabled on timer compare and timer disable</p>
3-2 Reserved	<p>This field is reserved.                      This read-only field is reserved and always has the value 0.</p>
1 TSTART	<p>Timer Start Bit</p> <p>When start bit is enabled, configured shifters will output the contents of the start bit when the timer is enabled and the timer counter will reload from the compare register on the first rising edge of the shift clock.</p> <p>0 Start bit disabled                      1 Start bit enabled</p>
0 Reserved	<p>This field is reserved.                      This read-only field is reserved and always has the value 0.</p>

**37.3.19 Timer Compare N Register (FLEXIO\_TIMCMPn)**

Address: 4005\_F000h base + 500h offset + (4d × i), where i=0d to 3d



**FLEXIO\_TIMCMPn field descriptions**

Field	Description
31-16 Reserved	<p>This field is reserved.                      This read-only field is reserved and always has the value 0.</p>
CMP	<p>Timer Compare Value</p> <p>The timer compare value is loaded into the timer counter when the timer is first enabled, when the timer is reset and when the timer decrements down to zero. In dual 8-bit counters baud/bit mode, the lower 8-bits configures the baud rate divider equal to (CMP[7:0] + 1) * 2. The upper 8-bits configure the number of bits in each word equal to (CMP[15:8] + 1) / 2. In dual 8-bit counters PWM mode, the lower 8-bits configure the high period of the output to (CMP[7:0] + 1) * 2. The upper 8-bits configure the low period of the output to (CMP[15:8] + 1) * 2. In 16-bit counter mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) to equal (CMP[15:0] + 1) * 2. When the shift clock source is a</p>

Table continues on the next page...

## FLEXIO\_TIMCMPn field descriptions (continued)

Field	Description
	pin or trigger input, the compare register is used to set the number of bits in each word equal to $(CMP[15:0] + 1) / 2$ .

## 37.4 Functional description

### 37.4.1 Shifter operation

Shifters are responsible for buffering and shifting data into or out of the FlexIO. The timing of shift, load and store events are controlled by the Timer assigned to the Shifter via the SHIFTCTL[TIMSEL] register. The Shifters are designed to support either DMA, interrupt or polled operation. The following block diagram provides a detailed view of the Shifter microarchitecture.

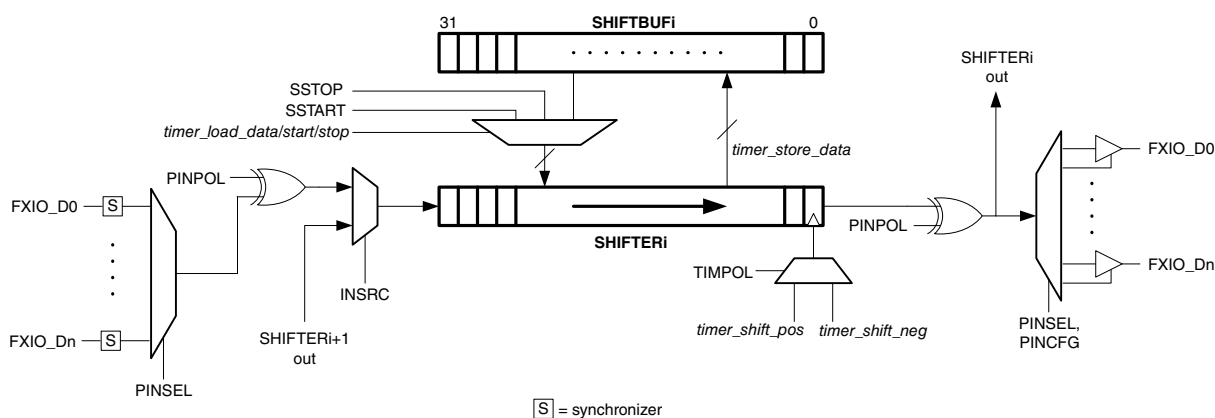


Figure 37-2. Shifter Microarchitecture

#### 37.4.1.1 Transmit Mode

When configured for Transmit mode (SHIFTCTL[SMOD]=Transmit), the shifter will load data from the SHIFTBUF register and shift data out when a load event is signalled by the assigned Timer. An optional start/stop bit can also be automatically loaded before/after SHIFTBUF data by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer. Note that the shifter will immediately load a stop bit when the Shifter is initially configured for Transmit mode if a stop bit is enabled.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when data has been loaded from the SHIFTBUF register into the Shifter or when the Shifter is initially configured into Transmit mode. The flag will clear when new data has been written into the SHIFTBUF register.

The Shifter Error Flag (SHIFTErr[SEF]) and any enabled interrupts will set when an attempt to load data from an empty SHIFTBUF register occurs (buffer underrun). The flag can be cleared by writing it with logic 1.

### **37.4.1.2 Receive Mode**

When configured for Receive mode (SHIFTCTL[SMOD]=Receive), the shifter will shift data in and store data into the SHIFTBUF register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when data has been stored into the SHIFTBUF register from the Shifter. The flag will clear when the data has been read from the SHIFTBUF register.

The Shifter Error Flag (SHIFTErr[SEF]) and any enabled interrupts will set when an attempt to store data into a full SHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

### **37.4.1.3 Match Store Mode**

When configured for Match Store mode (SHIFTCTL[SMOD]=Match Store), the shifter will shift data in, check for a match result and store matched data into the SHIFTBUF register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer. Up to 16-bits of data can be compared using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when a match occurs and matched data has been stored into the SHIFTBUF register from the Shifter. The flag will clear when the matched data has been read from the SHIFTBUF register.

The Shifter Error Flag (SHIFTErr[SEF]) and any enabled interrupts will set when an attempt to store matched data into a full SHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

#### 37.4.1.4 Match Continuous Mode

When configured for Match Continuous mode (SHIFTCTL[SMOD]=Match Continuous), the shifter will shift data in and continuously check for a match result whenever a shift event is signalled by the assigned Timer. Up to 16-bits of data can be compared using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set when a match occurs. The flag will clear automatically as soon as there is no longer a match between Shifter data and SHIFTBUF register.

The Shifter Error Flag (SHIFTErr[SEF]) and any enabled interrupts will set when a match occurs. The flag will clear when there is a read from the SHIFTBUF register or it written with logic 1.

### 37.4.2 Timer operation

The FlexIO 16-bit timers control the loading, shifting and storing of the shift registers, the counters load the contents of the compare register and decrement down to zero on the FlexIO clock. They can perform generic timer functions such as generating a clock or select output or a PWM waveform. Timers can be configured to enable in response to a trigger, pin or shifter condition; decrement always or only on a trigger or pin edge; reset in response to a trigger or pin condition; and disable on a trigger or pin condition or on a timer compare. Timers can optionally include a start condition and/or stop condition.

Each timer operates independently, although a timer can be configured to enable or disable at the same time as the previous timer (eg: timer1 can enable or disable at the same time as timer 0) and a timer output can be used to trigger any other timer. The trigger used by each timer is configured independently and can be configured to be a timer output, shifter status flag, pin input or an external trigger input (refer to the chip configuration section for details on the external trigger connections). The trigger configuration is separate from the pin configuration, which can be configured for input, output data or output enable.

The Timer Configuration Register (TIMCFGn) should be configured before setting the Timer Mode (TIMOD). Once the TIMOD is configured for the desired mode, when the condition configured by timer enable (TIMENA) is detected then the following events occur.

- Timer counter will load the current value of the Compare Register and start decrementing as configured by TIMDEC.
- Timer output will set depending on the TIMOUT configuration.
- Transmit shifters controlled by this timer will either output their start bit value, or load the shift register from the shift buffer and output the first bit, as configured by SSTART.

The Timer will then generate the timer output and timer shift clock depending on the TIMOD and TIMDEC fields. The shifter clock is either equal to the timer output (when TIMDEC=00 or 01) or equal to the decrement clock (when TIMDEC=10 or 11). When TIMDEC is configured to decrement from a pin or trigger, the timer will decrement on both rising and falling edges.

When the Timer is configured to reset as configured in the TIMRST field then the Timer counter will load the current value of the Compare Register again, the timer output may also be affected by the reset as configured in TIMOUT.

If the Timer start bit is enabled, the timer counter will reload with the compare register on the first rising edge of the shift clock after the timer starts decrementing. If there is no falling edge on the shift clock before the first rising edge (for example, when TIMOUT=1), a shifter that is configured to shift on falling edge and load on the first shift will not load correctly.

When configured for 8-bit counter mode, whenever the lower 8-bit counter decrements to zero the timer output will toggle, the lower 8-bit counter register will reload from the compare register and the upper 8-bit counter will decrement. For 8-bit PWM mode, the lower 8-bit counter will only decrement when the output is high and the upper 8-bit counter will only decrement when the output is low. The timer output will toggle whenever either lower or upper 8-bit counter decrements to zero.

When the timer decrements to zero, a compare event occurs depending on the timer mode. For 8-bit counter or PWM modes, both halves of the counter must equal zero and the upper half must decrement for the timer compare event to occur, while in 16-bit mode the entire counter must equal zero and decrement. The timer compare event will cause the timer status flag to set, the timer counter to load the contents of the timer compare register, the timer output to toggle, any configured transmit shift registers to load and any configured receive shift registers to store .

When the is Timer is configured to add a stop bit on each compare, the following additional events will occur.



- Transmit shifters controlled by this timer will output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer will store the contents of the shift register in their shift buffer, as configured by SSTOP.
- On the first rising edge of the shifter clock after the compare, the timer counter will reload the current value of the Compare Register.

Transmit shifters must be configured to load on the first shift when the timer is configured to insert a stop bit on each compare.

When the condition configured by timer disable (TIMDIS) is detected, the following events occur.

- Timer counter will reload the current value of the Compare Register and start decrementing as configured by TIMDEC.
- Timer output will clear.
- Transmit shifters controlled by this timer will output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer will store the contents of the shift register in their shift buffer, as configured by SSTOP.

If the timer stop bit is enabled, the timer counter will continue decrementing until the next rising edge of the shift clock is detected, at which point it will finish. A timer enable condition can be detected in the same cycle as a timer disable condition (if timer stop bit is disabled), or on the first rising edge of the shift clock after the disable condition (if stop bit is enabled). Receive shift registers will stop bit enabled will store the contents of the shift register into the shift buffer and verify the state of the input data on the configured shift edge while the timer is in the stop state condition. If there is no configured edge between the timer disable and the next rising edge of the shift clock then the final store and verify do not occur.

### 37.4.3 Pin operation

The pin configuration for each timer and shifter can be configured to use any FlexIO pin with either polarity. Each timer and shifter can be configured as an input, output data, output enable or bidirectional output. A pin configured for output enable can be used as an open drain (with inverted polarity, since the output enable assertion would cause logic zero to be output on the pin) or to control the enable on the bidirectional output. Any timer or shifter could be configured to control the output enable for a pin where the bidirectional output data is driven by another timer or shifter.

When configuring a pin as an input (this includes a timer trigger configured as a pin input), the input signal is first synchronized to the FlexIO clock before the signal is used by a timer or shifter. This introduces a small latency of between 0.5 to 1.5 FlexIO clock cycles when using an external pin input to generate an output or control a shifter. This sets the maximum setup time at 1.5 FlexIO clock cycles.

If an input is used by more than one timer or shifter then the synchronization occurs once to ensure any edge is seen on the same cycle by all timers and shifters using that input.

Note that FlexIO pins are also connected internally, configuring a FlexIO shifter or timer to output data on an unused pin will make an internal connection that allows other shifters and timer to use this pin as an input. This allows a shifter output to be used to trigger a timer or a timer output to be shifted into a shifter. This path is also synchronized to the FlexIO clock and therefore incurs a 1 cycle latency.

So when using a Pin input as a Timer Trigger, Timer Clock or Shifter Data Input, the following synchronization delays occur:

1. 0.5 – 1.5 FlexIO clock cycles for external pin
2. 1 FlexIO clock cycle for an internally driven pin

For timing considerations such as output valid time and input setup time for specific applications (SPI Master, SPI Slave, I2C Master, I2S Master, I2S Slave) please refer to the FlexIO Application Information Section.

## 37.5 Application Information

This section provides examples for a variety of FlexIO module applications.

### 37.5.1 UART Transmit

UART transmit can be supported using one Timer, one Shifter and one Pin (two Pins if supporting CTS). The start and stop bit insertion is handled automatically and multiple transfers can be supported using DMA controller. The timer status flag can be used to indicate when the stop bit of each word is transmitted.

Break and idle characters require software intervention, before transmitting a break or idle character the SSTART and SSTOP fields should be altered to transmit the required state and the data to transmit must equal 0xFF or 0x00. Supporting a second stop bit requires the stop bit to be inserted into the data stream using software (and increasing the

number of bits to transmit). Note that when performing byte writes to SHIFTBUF<sub>n</sub> (or SHIFTBUFBIS for transmitting MSB first), the rest of the register remains unaltered allowing an address mark bit or additional stop bit to remain undisturbed.

FlexIO does not support automatic insertion of parity bits.

**Table 37-3. UART Transmit Configuration**

Register	Value	Comments
SHIFTCFG <sub>n</sub>	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTL <sub>n</sub>	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0. Can invert output data by setting PINPOL, or can support open drain by setting PINPOL=0x1 and PINCFG=0x1.
TIMCMP <sub>n</sub>	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG <sub>n</sub>	0x0000_2222	Configure start bit, stop bit, enable on trigger low and disable on compare. Can support CTS by configuring TIMEN=0x3.
TIMCTL <sub>n</sub>	0x01C0_0001	Configure dual 8-bit counter using Shifter 0 status flag as inverted internal trigger source. Can support CTS by configuring PINSEL=0x1 (for Pin 1) and PINPOL=0x1.
SHIFTBUF <sub>n</sub>	Data to transmit	Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBIS[7:0] register instead.

## 37.5.2 UART Receive

UART receive can be supported using one Timer, one Shifter and one Pin (two Timers and two Pins if supporting RTS). The start and stop bit verification is handled automatically and multiple transfers can be supported using the DMA controller. The timer status flag can be used to indicate when the stop bit of each word is received.

Triple voting of the received data is not supported by FlexIO, data is sampled only once in the middle of each bit. Another timer can be used to implement a glitch filter on the incoming data, another Timer can also be used to detect an idle line of programmable length. Break characters will cause the error flag to set and the shifter buffer register will return 0x00.

FlexIO does not support automatic verification of parity bits.

**Table 37-4. UART Receiver Configuration**

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0080_0001	Configure receive using Timer 0 on negeedge of clock with input data on Pin 0. Can invert input data by setting PINPOL.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0000_2422	Configure start bit, stop bit, enable on pin posedge and disable on compare. Enable resynchronization to received data with TIMOUT=0x2 and TIMRST=0x4.
TIMCTLn	0x0000_0081	Configure dual 8-bit counter using inverted Pin 0 input.
SHIFTBUFn	Data to receive	Received data can be read from SHIFTBUF[31:24], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS[7:0] register instead.

The UART Receiver with RTS configuration uses a 2nd Timer to generate the RTS output. The RTS will assert when the start bit is detected and negate when the data is read from the shifter buffer register. No start bit will be detected while the RTS is asserted, the received data is simply ignored.

**Table 37-5. UART Receiver with RTS Configuration**

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0080_0001	Configure receive using Timer 0 on negeedge of clock with input data on Pin 0. Can invert input data by setting PINPOL.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0000_2522	Configure start bit, stop bit, enable on pin posedge with trigger low and disable on compare. Enable resynchronization

*Table continues on the next page...*

**Table 37-5. UART Receiver with RTS Configuration (continued)**

Register	Value	Comments
		to received data with TIMEOUT=0x2 and TIMRST=0x4.
TIMCTLn	0x03C0_0081	Configure dual 8-bit counter using inverted Pin 0 input. Trigger is internal using inverted Pin 1 input.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0030_6100	Enable on Timer N enable and disable on trigger falling edge. Decrement on trigger to ensure no compare.
TIMCTL(n+1)	0x0143_0083	Configure 16-bit counter and output on Pin 1. Trigger is internal using Shifter 0 flag.
SHIFTBUFn	Data to receive	Received data can be read from SHIFTBUF[31:24], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS[7:0] register instead.

### 37.5.3 SPI Master

SPI master mode can be supported using two Timers, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The stop bit is used to guarantee a minimum of 1 clock cycle between the slave select negating and before the next transfer. Writing to the transmit buffer by either core or DMA is used to initiate each transfer.

Due to synchronization delays, the setup time for the serial input data is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

**Table 37-6. SPI Master (CPHA=0) Configuration**

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFCTLn	0x0083_0002	Configure transmit using Timer 0 on negeedge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFCTL(n+1)	0x0000_0101	Configure receive using Timer 0 on posedge of clock with input data on Pin 1.

*Table continues on the next page...*

**Table 37-6. SPI Master (CPHA=0) Configuration (continued)**

Register	Value	Comments
TIMCMPn	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0100_2222	Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0. Set PINPOL to invert the output shift clock.
TIMCTLn	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0000_1100	Enable when Timer 0 is enabled and disable when Timer 0 is disabled.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBIS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUF, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

**Table 37-7. SPI Master (CPHA=1) Configuration**

Register	Value	Comments
SHIFTCFGn	0x0000_0021	Start bit loads data on first shift.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on negedge of clock with input data on Pin 1.
TIMCMPn	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.

Table continues on the next page...

**Table 37-7. SPI Master (CPHA=1) Configuration (continued)**

Register	Value	Comments
TIMCFGn	0x0100_2222	Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0. Set PINPOL to invert the output shift clock. Set TIMDIS=3 to keep slave select asserted for as long as there is data in the transmit buffer.
TIMCTLn	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0000_1100	Enable when Timer 0 is enabled and disable when Timer 0 is disabled.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBIS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUF, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

### 37.5.4 SPI Slave

SPI slave mode can be supported using one Timer, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The transmit data must be written to the transmit buffer register before the external slave select asserts, otherwise the shifter error flag will be set.

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

**Table 37-8. SPI Slave (CPHA=0) Configuration**

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFTCTLn	0x0083_0002	Configure transmit using Timer 0 on falling edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0000_0101	Configure receive using Timer 0 on rising edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_003F	Configure 32-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFGn	0x0120_6600	Configure enable on trigger rising edge and disable on trigger falling edge, initial clock state is logic 0 and decrement on pin input.
TIMCTLn	0x06C0_0203	Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBIS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUF, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

**Table 37-9. SPI Slave (CPHA=1) Configuration**

Register	Value	Comments
SHIFTCFGn	0x0000_0001	Shifter configured to load on first shift and stop bit disabled.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on rising edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on falling edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_003F	Configure 32-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFGn	0x0120_6602	Configure start bit, enable on trigger rising edge, disable on trigger falling edge, initial clock state is logic 0 and decrement on pin input.

Table continues on the next page...



**Table 37-9. SPI Slave (CPHA=1) Configuration (continued)**

Register	Value	Comments
TIMCTLn	0x06C0_0203	Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUF, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBS register instead.

### 37.5.5 I2C Master

I2C master mode can be supported using two Timers, two Shifters and two Pins. One timer is used to generate the SCL output and one timer is used to control the shifters. The two shifters are used to transmit and receive for every word, when receiving the transmitter must transmit 0xFF to tristate the output. FlexIO inserts a stop bit after every word to generate/verify the ACK/NACK. FlexIO waits for the first write to the transmit data buffer before enabling SCL generation. Data transfers can be supported using the DMA controller and the shifter error flag will set on transmit underrun or receive overflow.

The first timer generates the bit clock for the entire packet (START to Repeated START/STOP), so the compare register needs to be programmed with the total number of clock edges in the packet (minus one). The timer supports clock stretching using the reset counter when pin equal to output (although this increases both the clock high and clock low periods by at least 1 FlexIO clock cycle each). The second timer uses the SCL input pin to control the transmit/receive shift registers, this enforces an SDA data hold time by an extra 2 FlexIO clock cycles.

Both the transmit and receive shifters need to be serviced for each word in the transfer, the transmit shifter must transmit 0xFF when receiving and the receive shifter returns the data actually present on the SDA pin. The transmit shifter will load 1 additional word on the last falling edge of SCL pin, this word should be 0x00 if generating a STOP condition or 0xFF if generating a repeated START condition. During the last word of a master-receiver transfer, the transmit SSTOP bit should be set by software to generate a NACK.

The receive shift register will assert an error interrupt if a NACK is detected, but software is responsible for generating the STOP or repeated START condition. If a NACK is detected during master-transmit, the interrupt routine should immediately write the transmit shifter register with 0x00 (if generating STOP) or 0xFF (if generating repeated START). Software should then wait for the next rising edge on SCL and then disable both timers. The transmit shifter should then be disabled after waiting the setup delay for a repeated START or STOP condition.

Due to synchronization delays, the data valid time for the transmit output is 2 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The I2C master data valid is delayed 2 cycles because the clock output is passed through a synchronizer before clocking the transmit/receive shifter (to guarantee some SDA hold time). Since the SCL output is synchronous with FlexIO clock, the synchronization delay is 1 cycle and then 1 cycle to generate the output.

**Table 37-10. I2C Master Configuration**

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Start bit enabled (logic 0) and stop bit enabled (logic 1).
SHIFTCTLn	0x0101_0082	Configure transmit using Timer 1 on rising edge of clock with inverted output enable (open drain output) on Pin 0.
SHIFTCFG(n+1)	0x0000_0020	Start bit disabled and stop bit enabled (logic 0) for ACK/NACK detection.
SHIFTCTL(n+1)	0x0180_0001	Configure receive using Timer 1 on falling edge of clock with input data on Pin 0.
TIMCMPn	0x0000_2501	Configure 2 word transfer with baud rate of divide by 4 of the FlexIO clock. Set $TIMCMP[15:8] = (\text{number of words} \times 18) + 1$ . Set $TIMCMP[7:0] = (\text{baud rate divider} / 2) - 1$ .
TIMCFGn	0x0102_2222	Configure start bit, stop bit, enable on trigger high, disable on compare, reset if output equals pin. Initial clock state is logic 0 and is not affected by reset.
TIMCTLn	0x01C1_0101	Configure dual 8-bit counter using Pin 1 output enable (SCL open drain), with Shifter 0 flag as the inverted trigger.
TIMCMP(n+1)	0x0000_000F	Configure 8-bit transfer. Set $TIMCMP[15:0] = (\text{number of bits} \times 2) - 1$ .
TIMCFG(n+1)	0x0020_1112	Enable when Timer 0 is enabled, disable when Timer 0 is disabled, enable start bit and stop bit at end of each word, decrement on pin input.
TIMCTL(n+1)	0x01C0_0183	Configure 16-bit counter using inverted Pin 1 input (SCL).

Table continues on the next page...

**Table 37-10. I2C Master Configuration (continued)**

Register	Value	Comments
SHIFTBUF <sub>n</sub>	Data to transmit	Transmit data can be written to SHIFTBUFBBS[7:0], use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.
SHIFTBUF <sub>(n+1)</sub>	Data to receive	Received data can be read from SHIFTBUFBIS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

### 37.5.6 I2S Master

I2S master mode can be supported using two Timers, two Shifters and four Pins. One timer is used to generate the bit clock and control the shifters and one timer is used to generate the frame sync. FlexIO waits for the first write to the transmit data buffer before enabling bit clock and frame sync generation. Data transfers can be supported using the DMA controller and the shifter error flag will set on transmit underrun or receive overflow.

The bit clock frequency is an even integer divide of the FlexIO clock frequency, and the initial frame sync assertion occurs at the same time as the first bit clock edge. The timer uses the start bit to ensure the frame sync is generated one clock cycle before the first output data.

Due to synchronization delays, the setup time for the receiver input is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

**Table 37-11. I2S Master Configuration**

Register	Value	Comments
SHIFTCFG <sub>n</sub>	0x0000_0001	Load transmit data on first shift and stop bit disabled.
SHIFTCTL <sub>n</sub>	0x0003_0002	Configure transmit using Timer 0 on rising edge of clock with output data on Pin 0.
SHIFTCFG <sub>(n+1)</sub>	0x0000_0000	Start and stop bit disabled.
SHIFTCTL <sub>(n+1)</sub>	0x0080_0101	Configure receive using Timer 0 on falling edge of clock with input data on Pin 1.
TIMCMP <sub>n</sub>	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.

*Table continues on the next page...*

**Table 37-11. I2S Master Configuration (continued)**

Register	Value	Comments
TIMCFGn	0x0000_0202	Configure start bit, enable on trigger high and never disable. Initial clock state is logic 1.
TIMCTLn	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (bit clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert the output shift clock.
TIMCMP(n+1)	0x0000_007F	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:0] = (number of bits x baud rate divider) - 1.
TIMCFG(n+1)	0x0000_0100	Enable when Timer 0 is enabled and never disable.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter using inverted Pin 3 output (as frame sync).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support LSB first transfer by writing to SHIFTBUF register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead.

### 37.5.7 I2S Slave

I2S slave mode can be supported using two Timers, two Shifters and four Pins (for single transmit and single receive, other combinations of transmit and receive are possible).

The transmit data must be written to the transmit buffer register before the external frame sync asserts, otherwise the shifter error flag will be set.

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The output valid time of SPI slave is max 2.5 cycles because there is a maximum 1.5 cycle delay on the clock synchronization plus 1 cycle to output the data

Table 37-12. I2S Slave Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFTCTLn	0x0103_0002	Configure transmit using Timer 1 on rising edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0180_0101	Configure receive using Timer 1 on falling edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_007D	Configure two 32-bit transfers per frame. Set TIMCMP[15:0] = (number of bits x 4) - 3.
TIMCFGn	0x0030_2400	Configure enable on pin rising edge (inverted frame sync) and disable on compare, initial clock state is logic 1 and decrement on trigger input (bit clock).
TIMCTLn	0x0440_0383	Configure 16-bit counter using inverted Pin 3 input (frame sync), with Pin 2 input (bit clock) as the trigger.
TIMCMP(n+1)	0x0000_003F	Configure 32-bit transfers. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFG(n+1)	0x0020_3500	Configure enable on pin rising edge with trigger high and disable on compare with trigger low, initial clock state is logic 0 and decrement on pin input.
TIMCTL(n+1)	0x0340_0203	Configure 16-bit counter using Pin 2 input (bit clock), with Timer 0 output as the trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF <sub>BIS</sub> , use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support LSB first transfer by writing to SHIFTBUF register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUF <sub>BIS</sub> , use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead.



# Chapter 38

## General-Purpose Input/Output (GPIO)

### 38.1 Chip-specific GPIO information

#### 38.1.1 GPIO instantiation information

The device includes a number of pins, PTB0, PTB1, PTD7, and PTC3 with high current drive capability. These pins can be used to drive LED or power MOSFET directly. The high drive capability applies to all functions which are multiplexed on these pins (LPUART, TPM, SPI, I2C, CLK\_OUT...etc)

The drive capability of High current drive pins associated to I2C can be set in either I2C register or PCTRL register

##### 38.1.1.1 Pull devices and directions

The pull devices are enabled out of POR only on  $\overline{\text{RESET}}$ ,  $\overline{\text{NMI}}$  and respective SWD signals. Other pins can be enabled by writing to  $\text{PORTx\_PCRn[PE]}$ .

All the pins have controllable pull direction using the  $\text{PORTx\_PCRn[PS]}$  field. All the pins default to pullup except for SWD\_CLK, when enabled.

#### 38.1.2 GPIO accessibility in the memory map

The GPIO is multi-ported and can be accessed directly by the core with zero wait states at base address  $0xF800\_0000$ . It can also be accessed by the core through the cross bar/ AIPS interface at  $0x400F\_F000$  and at an aliased slot (15) at address  $0x4000\_F000$ . All

BME operations to the GPIO space can be accomplished referencing the aliased slot (15) at address 0x4000\_F000. Only some of the BME operations can be accomplished referencing GPIO at address 0x400F\_F000.

## 38.2 Introduction

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.

### 38.2.1 Features

Features of the GPIO module include:

- Port Data Input register visible in all digital pin-multiplexing modes
- Port Data Output register with corresponding set/clear/toggle registers
- Port Data Direction register
- Zero wait state access to GPIO registers through IOPORT

#### NOTE

The GPIO module is clocked by system clock.

### 38.2.2 Modes of operation

The following table depicts different modes of operation and the behavior of the GPIO module in these modes.

**Table 38-1. 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.



### 38.2.3 GPIO signal descriptions

Table 38-2. 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.

#### 38.2.3.1 Detailed signal description

Table 38-3. GPIO interface-detailed signal descriptions

Signal	I/O	Description		
PORTA31–PORTA0	I/O	General-purpose input/output		
PORTB31–PORTB0		State meaning	Asserted: The pin is logic 1. Deasserted: The pin is logic 0.	
PORTC31–PORTC0		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.	
PORTD31–PORTD0				
PORTE31–PORTE0				

#### 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.

### 38.3 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_PODR)	32	R/W	0000_0000h	<a href="#">38.3.1/739</a>
400F_F004	Port Set Output Register (GPIOA_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.2/740</a>
400F_F008	Port Clear Output Register (GPIOA_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.3/740</a>
400F_F00C	Port Toggle Output Register (GPIOA_OTDR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.4/741</a>
400F_F010	Port Data Input Register (GPIOA_PDIR)	32	R	0000_0000h	<a href="#">38.3.5/741</a>
400F_F014	Port Data Direction Register (GPIOA_PDDR)	32	R/W	0000_0000h	<a href="#">38.3.6/742</a>
400F_F040	Port Data Output Register (GPIOB_PODR)	32	R/W	0000_0000h	<a href="#">38.3.1/739</a>
400F_F044	Port Set Output Register (GPIOB_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.2/740</a>
400F_F048	Port Clear Output Register (GPIOB_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.3/740</a>
400F_F04C	Port Toggle Output Register (GPIOB_OTDR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.4/741</a>
400F_F050	Port Data Input Register (GPIOB_PDIR)	32	R	0000_0000h	<a href="#">38.3.5/741</a>
400F_F054	Port Data Direction Register (GPIOB_PDDR)	32	R/W	0000_0000h	<a href="#">38.3.6/742</a>
400F_F080	Port Data Output Register (GPIOC_PODR)	32	R/W	0000_0000h	<a href="#">38.3.1/739</a>
400F_F084	Port Set Output Register (GPIOC_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.2/740</a>
400F_F088	Port Clear Output Register (GPIOC_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.3/740</a>
400F_F08C	Port Toggle Output Register (GPIOC_OTDR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.4/741</a>
400F_F090	Port Data Input Register (GPIOC_PDIR)	32	R	0000_0000h	<a href="#">38.3.5/741</a>
400F_F094	Port Data Direction Register (GPIOC_PDDR)	32	R/W	0000_0000h	<a href="#">38.3.6/742</a>
400F_F0C0	Port Data Output Register (GPIOD_PODR)	32	R/W	0000_0000h	<a href="#">38.3.1/739</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_F0C4	Port Set Output Register (GPIOD_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.2/740</a>
400F_F0C8	Port Clear Output Register (GPIOD_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.3/740</a>
400F_F0CC	Port Toggle Output Register (GPIOD_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.4/741</a>
400F_F0D0	Port Data Input Register (GPIOD_PDIR)	32	R	0000_0000h	<a href="#">38.3.5/741</a>
400F_F0D4	Port Data Direction Register (GPIOD_PDDR)	32	R/W	0000_0000h	<a href="#">38.3.6/742</a>
400F_F100	Port Data Output Register (GPIOE_PDOR)	32	R/W	0000_0000h	<a href="#">38.3.1/739</a>
400F_F104	Port Set Output Register (GPIOE_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.2/740</a>
400F_F108	Port Clear Output Register (GPIOE_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.3/740</a>
400F_F10C	Port Toggle Output Register (GPIOE_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.3.4/741</a>
400F_F110	Port Data Input Register (GPIOE_PDIR)	32	R	0000_0000h	<a href="#">38.3.5/741</a>
400F_F114	Port Data Direction Register (GPIOE_PDDR)	32	R/W	0000_0000h	<a href="#">38.3.6/742</a>

### 38.3.1 Port Data Output Register (GPIOx\_PDOR)

This register configures the logic levels that are driven on each general-purpose output pins.

#### NOTE

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

Address: Base address + 0h offset

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

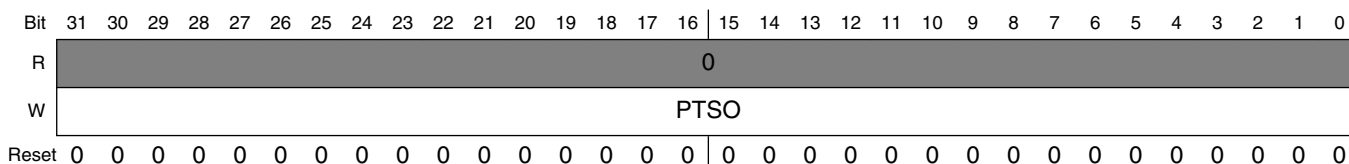
### GPIOx\_PDOR field descriptions

Field	Description
PDO	<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>

### 38.3.2 Port Set Output Register (GPIOx\_PSOR)

This register configures whether to set the fields of the PDOR.

Address: Base address + 4h offset



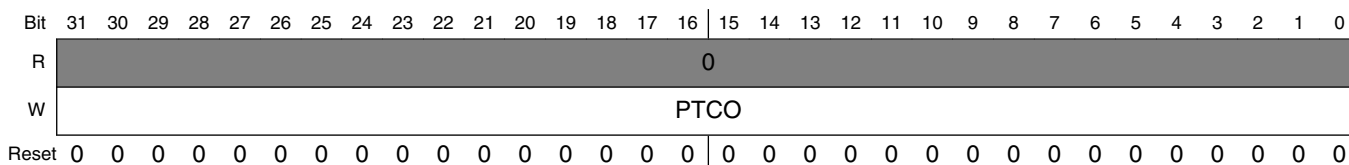
### GPIOx\_PSOR field descriptions

Field	Description
PTSO	<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>

### 38.3.3 Port Clear Output Register (GPIOx\_PCOR)

This register configures whether to clear the fields of PDOR.

Address: Base address + 8h offset



### GPIOx\_PCOR field descriptions

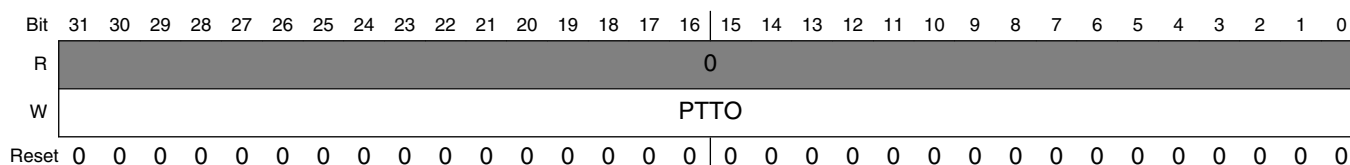
Field	Description
PTCO	Port Clear Output

**GPIOx\_PCOR field descriptions (continued)**

Field	Description
	Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows:  0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is cleared to logic 0.

**38.3.4 Port Toggle Output Register (GPIOx\_PTOR)**

Address: Base address + Ch offset

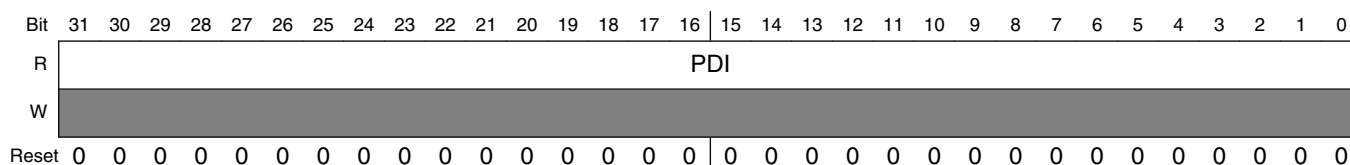
**GPIOx\_PTOR field descriptions**

Field	Description
PTTO	Port Toggle Output  Writing to this register will update the contents of the corresponding bit in the PDOR as follows:  0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to the inverse of its existing logic state.

**38.3.5 Port Data Input Register (GPIOx\_PDIR)****NOTE**

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

Address: Base address + 10h offset



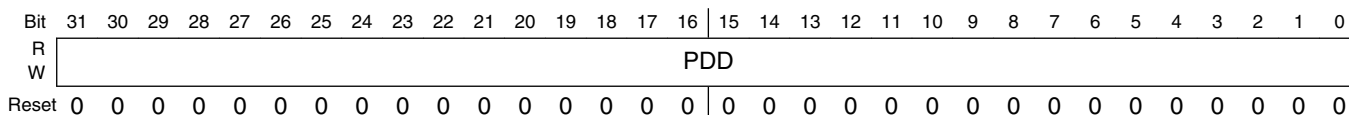
### GPIOx\_PDIR field descriptions

Field	Description
PDI	<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.                      1 Pin logic level is logic 1.</p>

### 38.3.6 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
PDD	<p>Port Data Direction</p> <p>Configures individual port pins for input or output.</p> <p>0 Pin is configured as general-purpose input, for the GPIO function.                      1 Pin is configured as general-purpose output, for the GPIO function.</p>

## 38.4 FGPIO memory map and register definition

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800\_0000.

Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. This aliased Fast GPIO memory map is called FGPIO.

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

## FGPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
F800_0000	Port Data Output Register (FGPIOA_PDOR)	32	R/W	0000_0000h	<a href="#">38.4.1/744</a>
F800_0004	Port Set Output Register (FGPIOA_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.2/745</a>
F800_0008	Port Clear Output Register (FGPIOA_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.3/745</a>
F800_000C	Port Toggle Output Register (FGPIOA_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.4/746</a>
F800_0010	Port Data Input Register (FGPIOA_PDIR)	32	R	0000_0000h	<a href="#">38.4.5/746</a>
F800_0014	Port Data Direction Register (FGPIOA_PDDR)	32	R/W	0000_0000h	<a href="#">38.4.6/747</a>
F800_0040	Port Data Output Register (FGPIOB_PDOR)	32	R/W	0000_0000h	<a href="#">38.4.1/744</a>
F800_0044	Port Set Output Register (FGPIOB_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.2/745</a>
F800_0048	Port Clear Output Register (FGPIOB_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.3/745</a>
F800_004C	Port Toggle Output Register (FGPIOB_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.4/746</a>
F800_0050	Port Data Input Register (FGPIOB_PDIR)	32	R	0000_0000h	<a href="#">38.4.5/746</a>
F800_0054	Port Data Direction Register (FGPIOB_PDDR)	32	R/W	0000_0000h	<a href="#">38.4.6/747</a>
F800_0080	Port Data Output Register (FGPIOC_PDOR)	32	R/W	0000_0000h	<a href="#">38.4.1/744</a>
F800_0084	Port Set Output Register (FGPIOC_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.2/745</a>
F800_0088	Port Clear Output Register (FGPIOC_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.3/745</a>
F800_008C	Port Toggle Output Register (FGPIOC_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.4/746</a>
F800_0090	Port Data Input Register (FGPIOC_PDIR)	32	R	0000_0000h	<a href="#">38.4.5/746</a>
F800_0094	Port Data Direction Register (FGPIOC_PDDR)	32	R/W	0000_0000h	<a href="#">38.4.6/747</a>
F800_00C0	Port Data Output Register (FGPIOD_PDOR)	32	R/W	0000_0000h	<a href="#">38.4.1/744</a>
F800_00C4	Port Set Output Register (FGPIOD_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.2/745</a>
F800_00C8	Port Clear Output Register (FGPIOD_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.3/745</a>

Table continues on the next page...

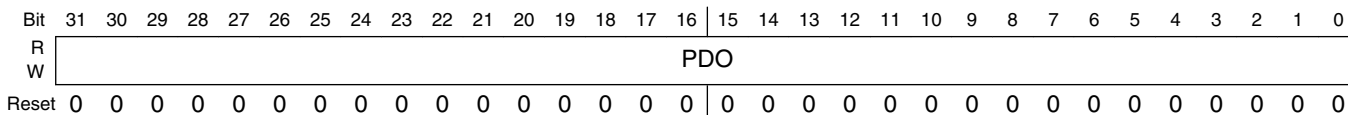
**FGPIO memory map (continued)**

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
F800_00CC	Port Toggle Output Register (FGPIOD_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.4/746</a>
F800_00D0	Port Data Input Register (FGPIOD_PDIR)	32	R	0000_0000h	<a href="#">38.4.5/746</a>
F800_00D4	Port Data Direction Register (FGPIOD_PDDR)	32	R/W	0000_0000h	<a href="#">38.4.6/747</a>
F800_0100	Port Data Output Register (FGPIOE_PDOR)	32	R/W	0000_0000h	<a href="#">38.4.1/744</a>
F800_0104	Port Set Output Register (FGPIOE_PSOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.2/745</a>
F800_0108	Port Clear Output Register (FGPIOE_PCOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.3/745</a>
F800_010C	Port Toggle Output Register (FGPIOE_PTOR)	32	W (always reads 0)	0000_0000h	<a href="#">38.4.4/746</a>
F800_0110	Port Data Input Register (FGPIOE_PDIR)	32	R	0000_0000h	<a href="#">38.4.5/746</a>
F800_0114	Port Data Direction Register (FGPIOE_PDDR)	32	R/W	0000_0000h	<a href="#">38.4.6/747</a>

**38.4.1 Port Data Output Register (FGPIOx\_PDOR)**

This register configures the logic levels that are driven on each general-purpose output pins.

Address: Base address + 0h offset



**FGPIOx\_PDOR field descriptions**

Field	Description
PDO	<p>Port Data Output</p> <p>Unimplemented pins for a particular device read as zero.</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>



## 38.4.2 Port Set Output Register (FGPIOx\_PSOR)

This register configures whether to set the fields of the PDOR.

Address: Base address + 4h offset

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

### FGPIOx\_PSOR field descriptions

Field	Description
PTSO	Port Set Output  Writing to this register will update the contents of the corresponding bit in the PDOR as follows:  0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to logic 1.

## 38.4.3 Port Clear Output Register (FGPIOx\_PCOR)

This register configures whether to clear the fields of PDOR.

Address: Base address + 8h offset

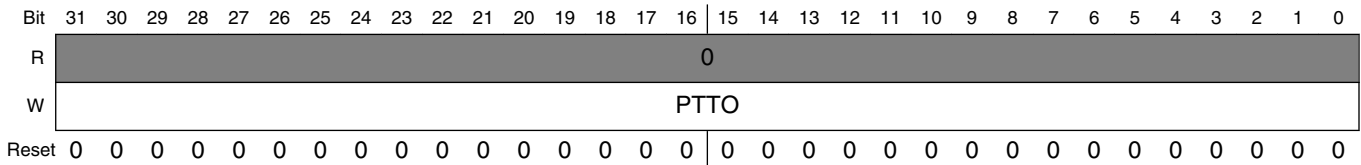
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																															
W	PTCO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### FGPIOx\_PCOR field descriptions

Field	Description
PTCO	Port Clear Output  Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows:  0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is cleared to logic 0.

### 38.4.4 Port Toggle Output Register (FGPIOx\_PTOR)

Address: Base address + Ch offset

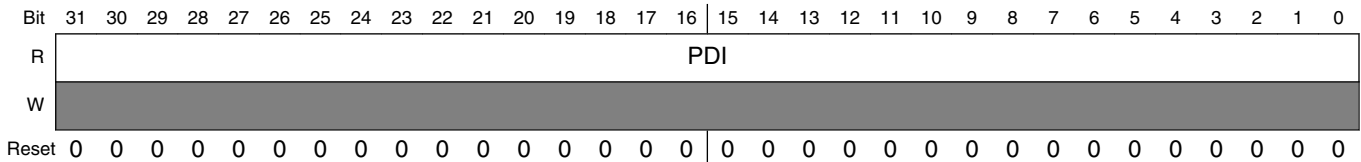


#### FGPIOx\_PTOR field descriptions

Field	Description
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>

### 38.4.5 Port Data Input Register (FGPIOx\_PDIR)

Address: Base address + 10h offset



#### FGPIOx\_PDIR field descriptions

Field	Description
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>

## 38.4.6 Port Data Direction Register (FGPIOx\_PDDR)

The PDDR configures the individual port pins for input or output.

Address: Base address + 14h offset

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

### FGPIOx\_PDDR field descriptions

Field	Description
PDD	<p>Port Data Direction</p> <p>Configures individual port pins for input or output.</p> <p>0 Pin is configured as general-purpose input, for the GPIO function.</p> <p>1 Pin is configured as general-purpose output, for the GPIO function.</p>

## 38.5 Functional description

### 38.5.1 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.

### 38.5.2 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
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.

### **38.5.3 IOPORT**

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800\_0000. Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. If the DMA attempts to access the GPIO registers on the same cycle as an IOPORT access, then the DMA access will stall until any IOPORT accesses have completed.

During Compute Operation, the GPIO registers remain accessible via the IOPORT interface only. Since the clocks to the Port Control and Interrupt modules are disabled during Compute Operation, the Pin Data Input Registers do not update with the current state of the pins.

# Chapter 39

## Bit Manipulation Engine (BME)

### 39.1 Introduction

The Bit Manipulation Engine (BME) provides hardware support for atomic read-modify-write memory operations to the peripheral address space in Cortex-M0+ based microcontrollers.

This architectural capability is also known as "decorated storage" as it defines a mechanism for providing additional semantics for load and store operations to memory-mapped peripherals beyond just the reading and writing of data values to the addressed memory locations. In the BME definition, the "decoration", that is, the additional semantic information, is encoded into the peripheral address used to reference the memory.

By combining the basic load and store instructions of the ARM Cortex-M instruction set architecture (v6M, v7M) with the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. The resulting architectural capability defined by this core platform function is targeted at the manipulation of n-bit fields in peripheral registers and is consistent with I/O hardware addressing in the Embedded C standard. For most BME commands, a single core read or write bus cycle is converted into an atomic read-modify-write, that is, an indivisible "read followed by a write" bus sequence.

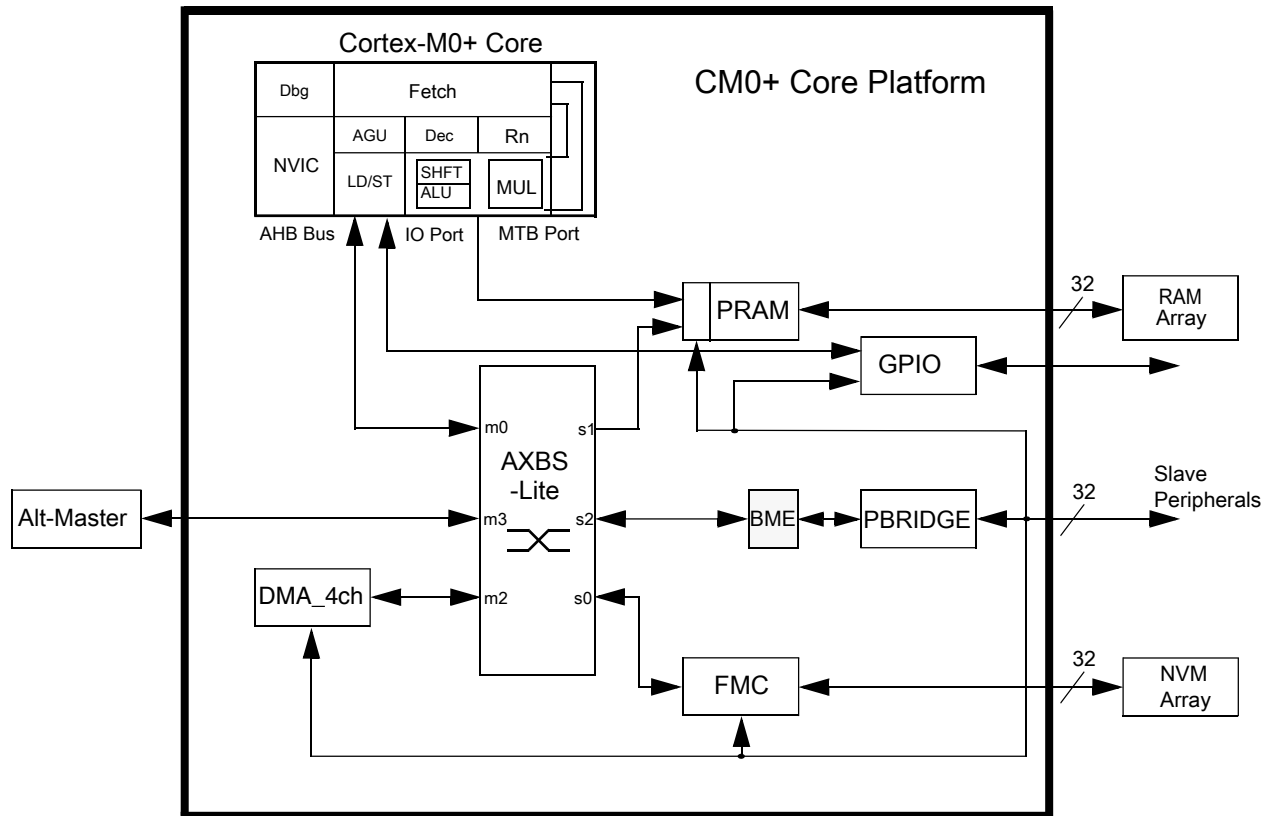
BME decorated references are only available on system bus transactions generated by the processor core and targeted at the standard 512 KB peripheral address space based at 0x4000\_0000<sup>1</sup>. The decoration semantic is embedded into address bits[28:19], creating a 448 MB space at addresses 0x4400\_0000–0x5FFF\_FFFF for AIPS; these bits are stripped out of the actual address sent to the peripheral bus controller and used by the BME to define and control its operation.

---

1. To be perfectly accurate, the peripheral address space occupies a 516 KB region: 512 KB based at 0x4000\_0000 plus a 4 KB space based at 0x400F\_F000 for GPIO accesses. This organization provides compatibility with the Kinetis K Family. Attempted accesses to the memory space located between 0x4008\_0000 - 0x400F\_EFFF are error terminated due to an illegal address.

### 39.1.1 Overview

The following figure is a generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers.



Note: BME can be accessed only by the core.

**Figure 39-1. Cortex-M0+ core platform block diagram**

As shown in the block diagram, the BME module interfaces to a crossbar switch AHB slave port as its primary input and sources an AHB bus output to the Peripheral Bridge (PBRIDGE) controller. The BME hardware microarchitecture is a 2-stage pipeline design matching the protocol of the AMBA-AHB system bus interfaces. The PBRIDGE module converts the AHB system bus protocol into the IPS/APB protocol used by the attached slave peripherals.

### 39.1.2 Features

The key features of the BME include:

- Lightweight implementation of decorated storage for selected address spaces

- Additional access semantics encoded into the reference address
- Resides between a crossbar switch slave port and a peripheral bridge bus controller
- Two-stage pipeline design matching the AHB system bus protocol
- Combinationally passes non-decorated accesses to peripheral bridge bus controller
- Conversion of decorated loads and stores from processor core into atomic read-modify-writes
- Decorated loads support unsigned bit field extracts, load-and-`{set,clear}` 1-bit operations
- Decorated stores support bit field inserts, logical AND, OR, and XOR operations
- Support for byte, halfword and word-sized decorated operations
- Supports minimum signal toggling on AHB output bus to reduce power dissipation

### 39.1.3 Modes of operation

The BME module does not support any special modes of operation. As a memory-mapped device located on a crossbar slave AHB system bus port, BME responds strictly on the basis of memory addresses for accesses to the peripheral bridge bus controller.

All functionality associated with the BME module resides in the core platform's clock domain; this includes its connections with the crossbar slave port and the PBRIDGE bus controller.

## 39.2 Memory map and register definition

The BME module provides a memory-mapped capability and does not include any programming model registers.

The exact set of functions supported by the BME are detailed in the [Functional description](#).

The peripheral address space occupies a 516 KB region: 512 KB based at 0x4000\_0000 plus a 4 KB space based at 0x400F\_F000 for GPIO accesses; the decorated address space is mapped to the 448 MB region located at 0x4400\_0000–0x5FFF\_FFFF.

## 39.3 Functional description

Information found here details the specific functions supported by the BME.

Recall the combination of the basic load and store instructions of the Cortex-M instruction set architecture (v6M, v7M) plus the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. The resulting architectural capability defined by this core platform function is targeted at the manipulation of n-bit fields in peripheral registers and is consistent with I/O hardware addressing in the Embedded C standard. For most BME commands, a single core read or write bus cycle is converted into an atomic read-modify-write, that is, an indivisible "read followed by a write" bus sequence.

Consider decorated store operations first, then decorated loads.

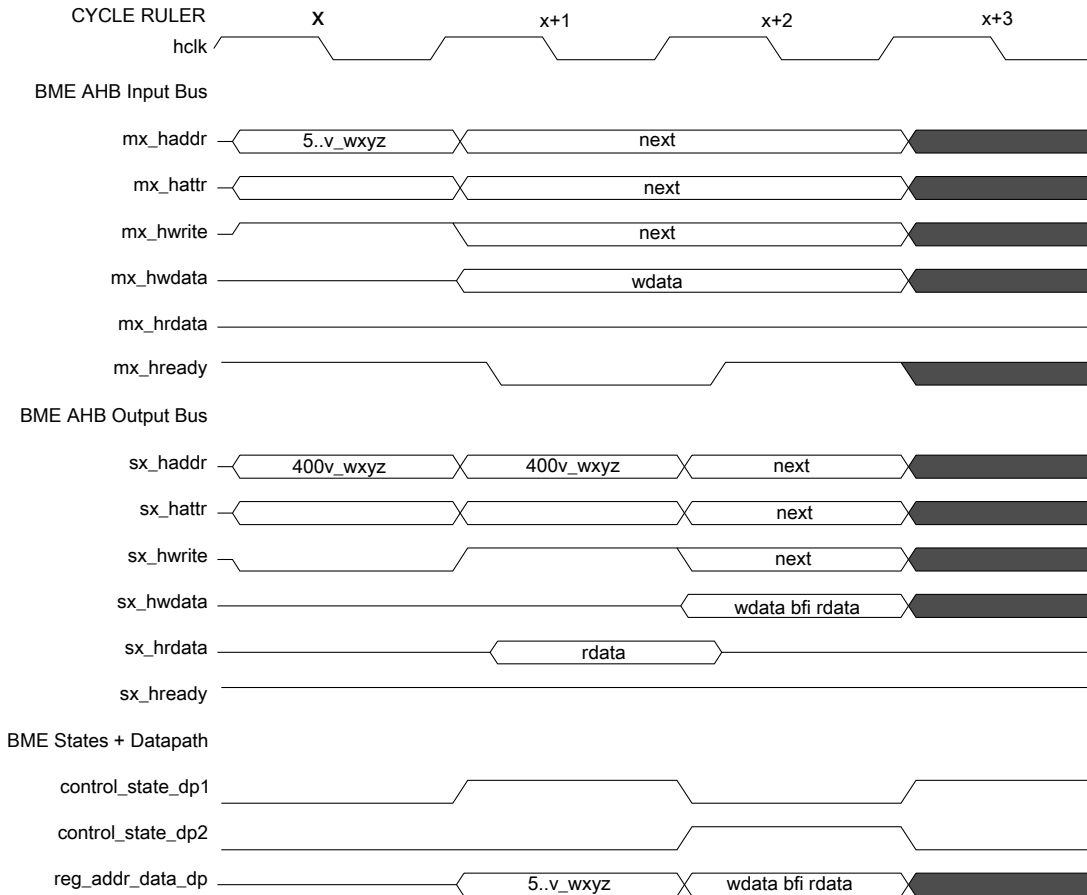
### **39.3.1 BME decorated stores**

The functions supported by the BME's decorated stores include three logical operators (AND, OR, XOR) plus a bit field insert.

For all these operations, BME converts a single decorated AHB store transaction into a 2-cycle atomic read-modify-write sequence, where the combined read-modify operation is performed in the first AHB data phase, and then the write is performed in the second AHB data phase.

A generic timing diagram of a decorated store showing a peripheral bit field insert operation is shown as follows:





**Figure 39-2. Decorated store: bit field insert timing diagram**

All the decorated store operations follow the same execution template shown in [Figure 39-2](#), a two-cycle read-modify-write operation:

1. Cycle x, 1st AHB address phase: Write from input bus is translated into a read operation on the output bus using the actual memory address (with the decoration removed) and then captured in a register.
2. Cycle x+1, 2nd AHB address phase: Write access with the registered (but actual) memory address is output
3. Cycle x+1, 1st AHB data phase: Memory read data is modified using the input bus write data and the function defined by the decoration and captured in a data register; the input bus cycle is stalled.
4. Cycle x+2, 2nd AHB data phase: Registered write data is sourced onto the output write data bus.

#### NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

### 39.3.1.1 Decorated store logical AND (AND)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read;
2. It is then modified by performing a logical AND operation using the write data operand sourced for the system bus cycle
3. Finally, the result of the AND operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

	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
ioandb	0	*	*	0	0	1	-	-	-	-	-	-	mem_addr																			
ioandh	0	*	*	0	0	1	-	-	-	-	-	-	mem_addr															0				
ioandw	0	*	*	0	0	1	-	-	-	-	-	-	mem_addr															0	0			

**Figure 39-3. Decorated store address: logical AND**

See [Figure 39-3](#) where `addr[30:29] = 10` for peripheral, `addr[28:26] = 001` specifies the AND operation, and `mem_addr[19:0]` specifies the address offset into the space based at `0x4000_0000` for peripherals. The "-" indicates an address bit "don't care".

The decorated AND write operation is defined in the following pseudo-code as:

```
ioand<sz>(accessAddress, wdata)           // decorated store AND
tmp = mem[accessAddress & 0xE0FFFFFF, size] // memory read
tmp = tmp & wdata                          // modify
mem[accessAddress & 0xE0FFFFFF, size] = tmp // memory write
```

where the operand size `<sz>` is defined as `b`(yte, 8-bit), `h`(alfword, 16-bit) and `w`(ord, 32-bit). This notation is used throughout the document.

In the cycle definition tables, the notations `AHB_ap` and `AHB_dp` refer to the address and data phases of the BME AHB transaction. The cycle-by-cycle BME operations are detailed in the following table.

**Table 39-1. Cycle definitions of decorated store: logical AND**

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert	Recirculate captured addr + attr to memory as slave_wt	<next>

*Table continues on the next page...*

**Table 39-1. Cycle definitions of decorated store: logical AND (continued)**

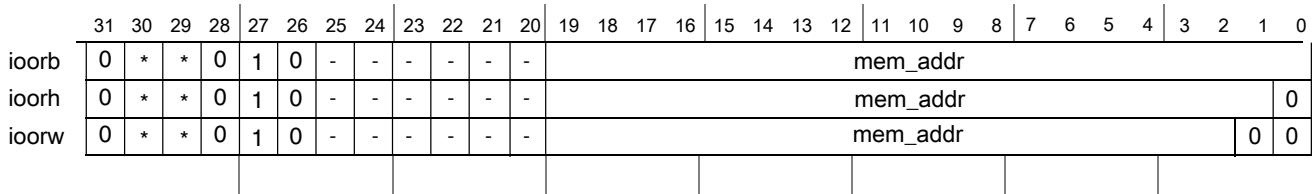
Pipeline stage	Cycle		
	x	x+1	x+2
	master_wt to slave_rd; Capture address, attributes		
BME AHB_dp	<previous>	Perform memory read; Form (rdata & wdata) and capture destination data in register	Perform write sending registered data to memory

### 39.3.1.2 Decorated store logical OR (OR)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read.
2. It is then modified by performing a logical OR operation using the write data operand sourced for the system bus cycle.
3. Finally, the result of the OR operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.



**Figure 39-4. Decorated address store: logical OR**

See [Figure 39-4](#), where `addr[30:29] = 10` for peripheral, `addr[28:26] = 010` specifies the OR operation, and `mem_addr[19:0]` specifies the address offset into the space based at `0x4000_0000` for peripherals. The "-" indicates an address bit "don't care".

The decorated OR write operation is defined in the following pseudo-code as:

```
ioor<sz>(accessAddress, wdata) // decorated store OR

tmp = mem[accessAddress & 0xE0FFFFFF, size] // memory read
tmp = tmp | wdata // modify
mem[accessAddress & 0xE0FFFFFF, size] = tmp // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

**Table 39-2. Cycle definitions of decorated store: logical OR**

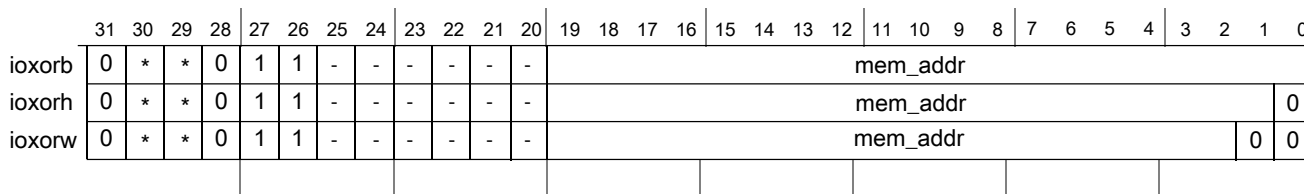
Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form (rdata   wdata) and capture destination data in register	Perform write sending registered data to memory

### 39.3.1.3 Decorated store logical XOR (XOR)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read.
2. It is then modified by performing a logical XOR (exclusive-OR) operation using the write data operand sourced for the system bus cycle.
3. Finally, the result of the XOR operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.



**Figure 39-5. Decorated address store: logical XOR**

See [Figure 39-5](#), where `addr[30:29] = 10` for peripheral, `addr[28:26] = 011` specifies the XOR operation, and `mem_addr[19:0]` specifies the address offset into the peripheral space based at `0x4000_0000` for peripherals. The "-" indicates an address bit "don't care".

The decorated XOR write operation is defined in the following pseudo-code as:

```
ioxor<sz>(accessAddress, wdata) // decorated store XOR

tmp = mem[accessAddress & 0xE00FFFFFF, size] // memory read
tmp = tmp ^ wdata // modify
mem[accessAddress & 0xE00FFFFFF, size] = tmp // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

**Table 39-3. Cycle definitions of decorated store: logical XOR**

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form (rdata ^ wdata) and capture destination data in register	Perform write sending registered data to memory

### 39.3.1.4 Decorated store bit field insert (BFI)

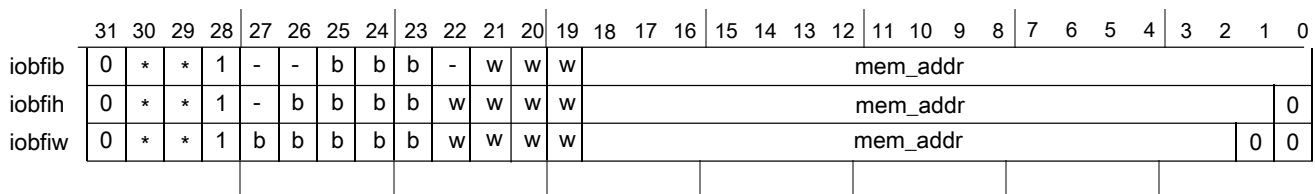
This command inserts a bit field contained in the write data operand, defined by LSB position (b) and the bit field width (w+1), into the memory "container" defined by the access size associated with the store instruction using an atomic read-modify-write sequence.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

#### NOTE

For the word sized operation, the maximum bit field width is 16 bits. The core performs the required write data lane replication on byte and halfword transfers.

The BFI operation can be used to insert a single bit into a peripheral. For this case, the w field is simply set to 0, indicating a bit field width of 1.

**Figure 39-6. Decorated address store: bit field insert**

where  $\text{addr}[30:29] = 10$  for peripheral,  $\text{addr}[28] = 1$  signals a BFI operation,  $\text{addr}[27:23]$  is "b", the LSB identifier,  $\text{addr}[22:19]$  is "w", the bit field width minus 1 identifier, and  $\text{addr}[18:0]$  specifies the address offset into the peripheral space based at  $0x4000\_0000$  for peripherals. The "-" indicates an address bit "don't care". Note, unlike the other decorated store operations, BFI uses  $\text{addr}[19]$  as the least significant bit in the "w" specifier and not as an address bit.

## Functional description

The decorated BFI write operation is defined in the following pseudo-code as:

```
iobfi<sz>(accessAddress, wdata)           // decorated bit field insert

tmp   = mem[accessAddress & 0xE007FFFF, size] // memory read
mask  = ((1 << (w+1)) - 1) << b             // generate bit mask
tmp   = tmp & ~mask                         // modify
      | wdata & mask
mem[accessAddress & 0xE007FFFF, size] = tmp // memory write
```

The write data operand (wdata) associated with the store instruction contains the bit field to be inserted. It must be properly aligned within a right-aligned container, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword, or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field "xyz" into an 8-bit memory container, initially set to "abcd\_efgh". For all cases, w is 2, signaling a bit field width of 3.

```
if b = 0 and the decorated store (strb) Rt register[7:0] = ----_-xyz,
    then destination is "abcd_xyz"
if b = 1 and the decorated store (strb) Rt register[7:0] = ----_xyz-,
    then destination is "abcd_xyzh"
if b = 2 and the decorated store (strb) Rt register[7:0] = ---x_ymz--,
    then destination is "abcx_ymzgh"
if b = 3 and the decorated store (strb) Rt register[7:0] = --xy_z---,
    then destination is "abxy_zfgh"
if b = 4 and the decorated store (strb) Rt register[7:0] = -xyz_----,
    then destination is "axyz_efgh"
if b = 5 and the decorated store (strb) Rt register[7:0] = xyz-____,
    then destination is "xyzd_efgh"
if b = 6 and the decorated store (strb) Rt register[7:0] = yz--____,
    then destination is "yzcd_efgh"
if b = 7 and the decorated store (strb) Rt register[7:0] = z---____,
    then destination is "zbcd_efgh"
```

Note from the example, when the starting bit position plus the field width exceeds the container size, only part of the source bit field is inserted into the destination memory location. Stated differently, if  $(b + w + 1) > \text{container\_width}$ , only the low-order "container\_width - b" bits are actually inserted.

The cycle-by-cycle BME operations are detailed in the following table.

**Table 39-4. Cycle definitions of decorated store: bit field insert**

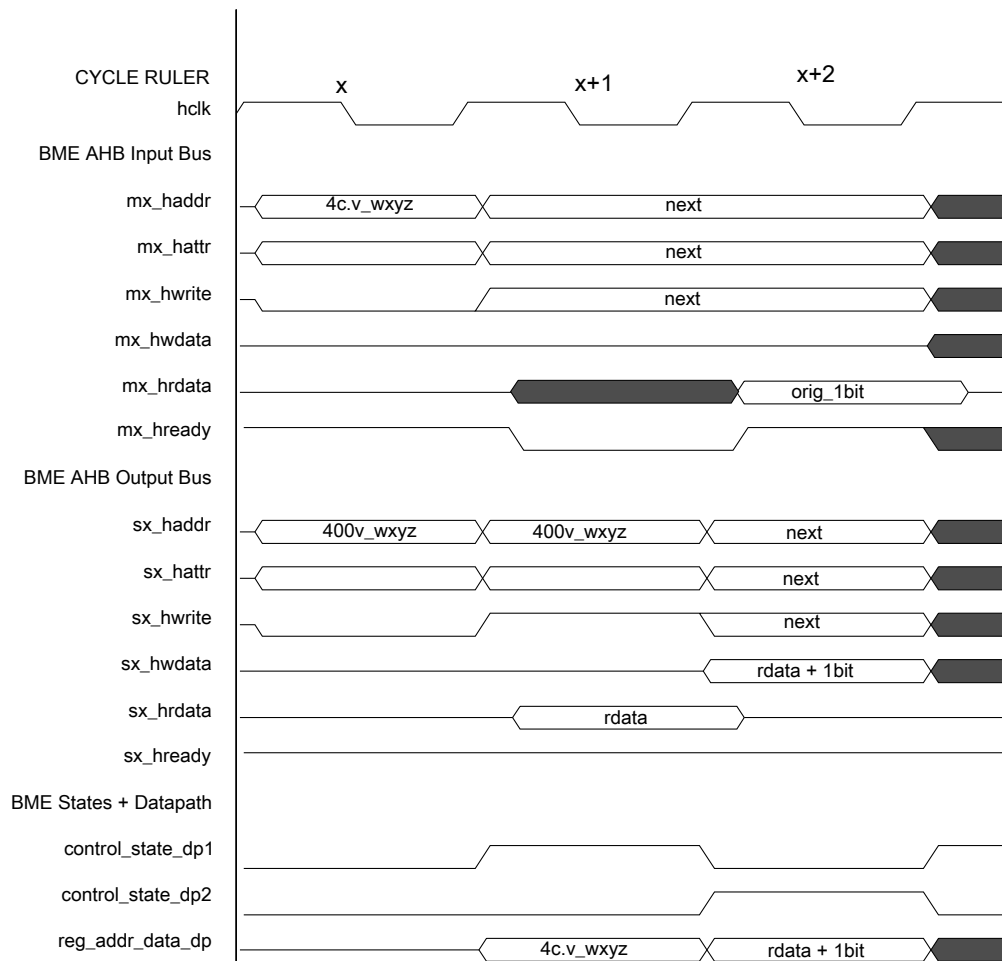
Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Form bitwise $((\text{mask}) ? \text{wdata} : \text{rdata})$ and capture destination data in register	Perform write sending registered data to memory

### 39.3.2 BME decorated loads

The functions supported by the BME's decorated loads include two single-bit load-and-`{set, clear}` operators plus unsigned bit field extracts.

For the two load-and-`{set, clear}` operations, BME converts a single decorated AHB load transaction into a two-cycle atomic read-modify-write sequence, where the combined read-modify operations are performed in the first AHB data phase, and then the write is performed in the second AHB data phase as the original read data is returned to the processor core. For an unsigned bit field extract, the decorated load transaction is stalled for one cycle in the BME as the data field is extracted, then aligned and returned to the processor in the second AHB data phase. This is the only decorated transaction that is not an atomic read-modify-write, as it is a simple data read.

A generic timing diagram of a decorated load showing a peripheral load-and-set 1-bit operation is shown as follows.



**Figure 39-7. Decorated load: load-and-set 1-bit field insert timing diagram**

Decorated load-and-`{set, clear}` 1-bit operations follow the execution template shown in the above figure: a 2-cycle read-modify-write operation:

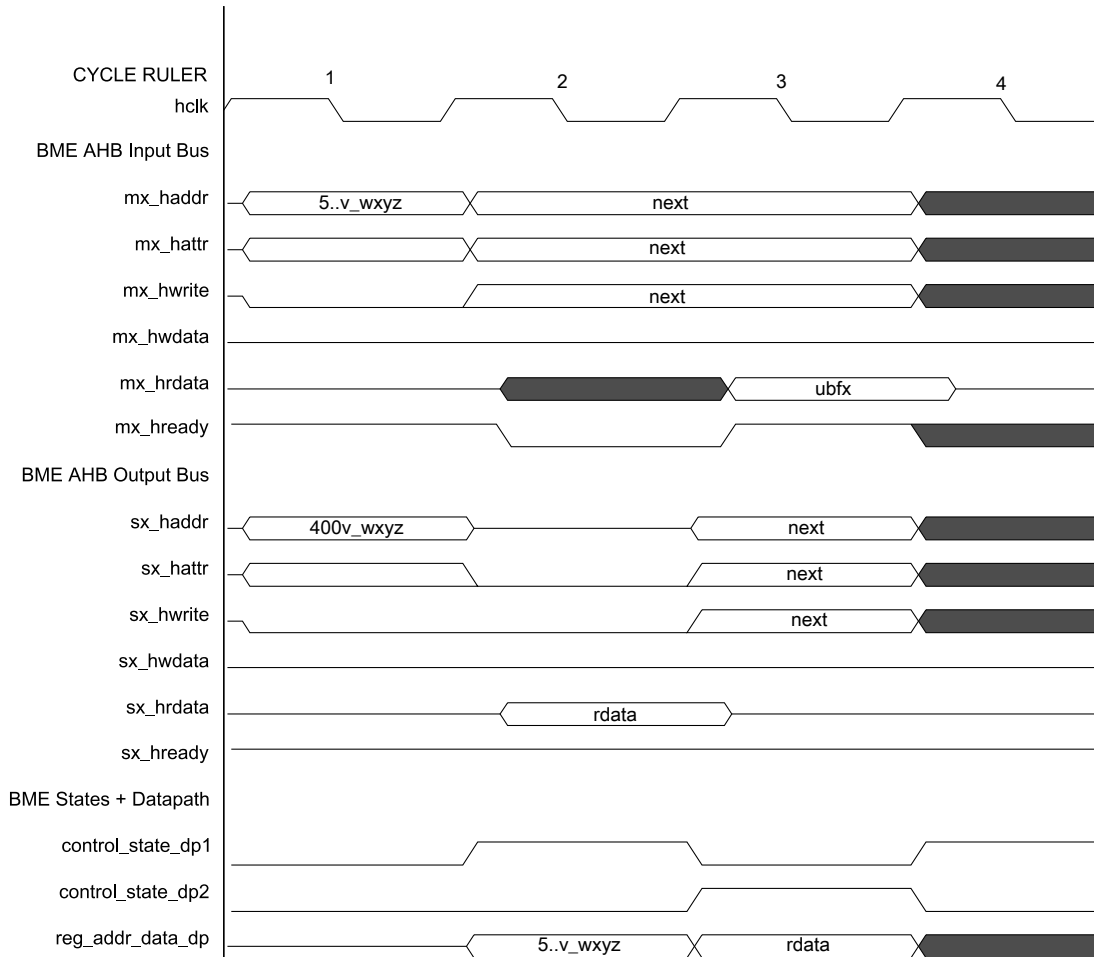
1. Cycle x, first AHB address phase: Read from input bus is translated into a read operation on the output bus with the actual memory address (with the decoration removed) and then captured in a register
2. Cycle x+1, second AHB address phase: Write access with the registered (but actual) memory address is output
3. Cycle x+1, first AHB data phase: The "original" 1-bit memory read data is captured in a register, while the 1-bit field is set or clear based on the function defined by the decoration with the modified data captured in a register; the input bus cycle is stalled
4. Cycle x+2, second AHB data phase: The selected original 1-bit is right-justified, zero-filled and then driven onto the input read data bus, while the registered write data is sourced onto the output write data bus



**NOTE**

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

A generic timing diagram of a decorated load showing an unsigned peripheral bit field operation is shown in the following figure.



**Figure 39-8. Decorated load: unsigned bit field insert timing diagram**

The decorated unsigned bit field extract follows the same execution template shown in the above figure, a 2-cycle read operation:

- Cycle x, 1st AHB address phase: Read from input bus is translated into a read operation on the output bus with the actual memory address (with the decoration removed) and then captured in a register
- Cycle x+1, 2nd AHB address phase: Idle cycle

## Functional description

- Cycle x+1, 1st AHB data phase: A bit mask is generated based on the starting bit position and the field width; the mask is AND'ed with the memory read data to isolate the bit field; the resulting data is captured in a data register; the input bus cycle is stalled
- Cycle x+2, 2nd AHB data phase: Registered data is logically right-aligned for proper alignment and driven onto the input read data bus

### NOTE

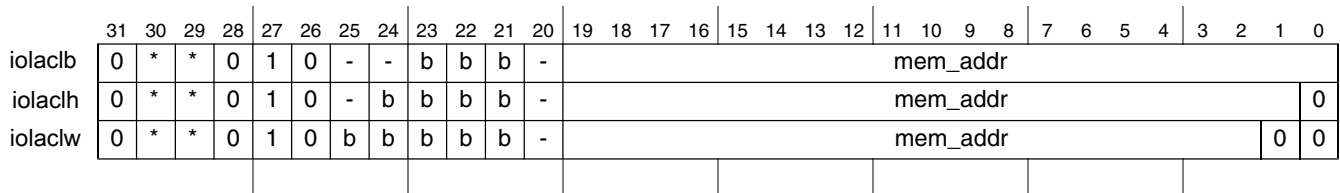
Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

### 39.3.2.1 Decorated load: load-and-clear 1 bit (LAC1)

This command loads a 1-bit field defined by the LSB position (b) into the core's general purpose destination register (Rt) and zeroes the bit in the memory space after performing an atomic read-modify-write sequence.

The extracted 1-bit data field from the memory address is right-justified and zero-filled in the operand returned to the core.

The data size is specified by the read operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).



**Figure 39-9. Decorated load address: load-and-clear 1 bit**

See [Figure 39-9](#) where  $\text{addr}[30:29] = 10$  for peripheral,  $\text{addr}[28:26] = 010$  specifies the load-and-clear 1 bit operation,  $\text{addr}[25:21]$  is "b", the bit identifier, and  $\text{mem\_addr}[19:0]$  specifies the address offset into the space based at  $0x4000\_0000$  for peripheral. The "-" indicates an address bit "don't care".

The decorated load-and-clear 1-bit read operation is defined in the following pseudo-code as:

```

rdata = ioacl<sz>(accessAddress)           // decorated load-and-clear 1
tmp    = mem[accessAddress & 0xE00FFFFF, size] // memory read
mask   = 1 << b                               // generate bit mask
rdata  = (tmp & mask) >> b                     // read data returned to core
tmp    = tmp & ~mask                           // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp   // memory write

```

The cycle-by-cycle BME operations are detailed in the following table.

**Table 39-5. Cycle definitions of decorated load: load-and-clear 1 bit**

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Extract bit from rdata; Form (rdata & ~mask) and capture destination data in register	Return extracted bit to master; Perform write sending registered data to memory

### 39.3.2.2 Decorated Load: Load-and-Set 1 Bit (LAS1)

This command loads a 1-bit field defined by the LSB position (b) into the core's general purpose destination register (Rt) and sets the bit in the memory space after performing an atomic read-modify-write sequence.

The extracted one bit data field from the memory address is right justified and zero filled in the operand returned to the core.

The data size is specified by the read operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

	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													
iolasb	0	*	*	0	1	1	-	-	b	b	b	-	mem_addr																																
iolash	0	*	*	0	1	1	-	b	b	b	b	-	mem_addr																															0	
iolasw	0	*	*	0	1	1	b	b	b	b	b	-	mem_addr																															0	0

**Figure 39-10. Decorated load address: load-and-set 1 bit**

where  $\text{addr}[30:29] = 10$  for peripheral,  $\text{addr}[28:26] = 011$  specifies the load-and-set 1 bit operation,  $\text{addr}[25:21]$  is "b", the bit identifier, and  $\text{mem\_addr}[19:0]$  specifies the address offset into the space based at  $0x4000\_0000$  for peripheral. The "-" indicates an address bit "don't care".

The decorated Load-and-Set 1 Bit read operation is defined in the following pseudo-code as:

```

rdata = iolas1<sz>(accessAddress)           // decorated load-and-set 1

tmp    = mem[accessAddress & 0xE00FFFFFFF, size] // memory read
mask   = 1 << b                                 // generate bit mask
rdata  = (tmp & mask) >> b                     // read data returned to core

```

## Functional description

```
tmp = tmp | mask // modify
mem[accessAddress & 0xE0FFFFFF, size] = tmp // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

**Table 39-6. Cycle definitions of decorated load: load-and-set 1-bit**

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Extract bit from rdata; Form (rdata   mask) and capture destination data in register	Return extracted bit to master; Perform write sending registered data to memory

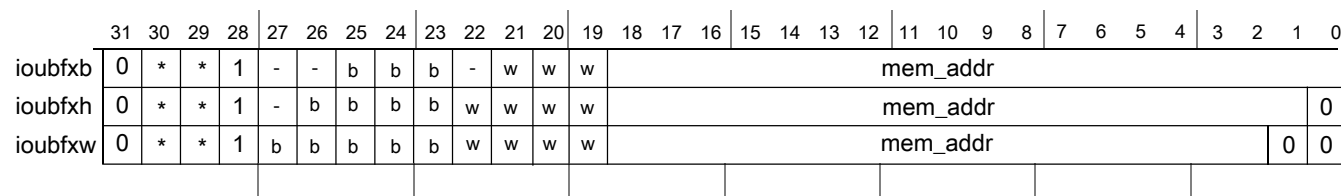
### 39.3.2.3 Decorated load unsigned bit field extract (UBFX)

This command extracts a bit field defined by LSB position (b) and the bit field width (w +1) from the memory "container" defined by the access size associated with the load instruction using a two-cycle read sequence.

The extracted bit field from the memory address is right-justified and zero-filled in the operand returned to the core. Recall this is the only decorated operation that does not perform a memory write, that is, UBFX only performs a read.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). Note for the word sized operation, the maximum bit field width is 16 bits.

The use of a UBFX operation is recommended to extract a single bit. For this case, the w field is simply set to 0, indicating a bit field width of 1.



**Figure 39-11. Decorated load address: unsigned bit field extract**

See [Figure 39-11](#), where  $\text{addr}[30:29] = 10$  for peripheral,  $\text{addr}[28] = 1$  specifies the unsigned bit field extract operation,  $\text{addr}[27:23]$  is "b", the LSB identifier,  $\text{addr}[22:19]$  is "w", the bit field width minus 1 identifier, and  $\text{mem\_addr}[18:0]$  specifies the address

offset into the space based at 0x4000\_0000 for peripheral. The "-" indicates an address bit "don't care". Note, unlike the other decorated load operations, UBFX uses addr[19] as the least significant bit in the "w" specifier and not as an address bit.

The decorated unsigned bit field extract read operation is defined in the following pseudo-code as:

```
rdata = ioubfx<sz>(accessAddress)           // unsigned bit field extract

tmp    = mem[accessAddress & 0xE007FFFF, size] // memory read
mask   = ((1 << (w+1)) - 1) << b             // generate bit mask
rdata  = (tmp & mask) >> b                   // read data returned to core
```

Like the BFI operation, when the starting bit position plus the field width exceeds the container size, only part of the source bit field is extracted from the destination memory location. Stated differently, if  $(b + w + 1) > \text{container\_width}$ , only the low-order "container\_width - b" bits are actually extracted. The cycle-by-cycle BME operations are detailed in the following table.

**Table 39-7. Cycle definitions of decorated load: unsigned bit field extract**

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Idle AHB address phase	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Form (rdata & mask) and capture destination data in register	Logically right shift registered data; Return justified rdata to master

### 39.3.3 Additional details on decorated addresses and GPIO accesses

As previously noted, the peripheral address space occupies a 516 KB region: 512 KB based at 0x4000\_0000 plus a 4 KB space based at 0x400F\_F000 for GPIO accesses. This memory layout provides compatibility with the Kinetis K Family and provides 129 address "slots", each 4 KB in size.

The GPIO address space is multiply-mapped by the hardware: it appears at the "standard" system address 0x400F\_F000 and is physically located in the address slot corresponding to address 0x4000\_F000. Decorated loads and stores create a slight complication involving accesses to the GPIO. Recall the use of address[19] varies by decorated operation; for AND, OR, XOR, LAC1 and LAS1, this bit functions as a true address bit, while for BFI and UBFX, this bit defines the least significant bit of the "w" bit field specifier.

## Application information

As a result, undecorated GPIO references and decorated AND, OR, XOR, LAC1 and LAS1 operations can use the standard 0x400F\_F000 base address, while decorated BFI and UBFX operations must use the alternate 0x4000\_F000 base address. Another implementation can simply use 0x400F\_F000 as the base address for all undecorated GPIO accesses and 0x4000\_F000 as the base address for all decorated accesses. Both implementations are supported by the hardware.

**Table 39-8. Decorated peripheral and GPIO address details**

Peripheral address space	Description
0x4000_0000–0x4007_FFFF	Undecorated (normal) peripheral accesses
0x4008_0000–0x400F_EFFF	Illegal addresses; attempted references are aborted and error terminated
0x400F_F000–0x400F_FFFF	Undecorated (normal) GPIO accesses using standard address
0x4010_0000–0x43FF_FFFF	Illegal addresses; attempted references are aborted and error terminated
0x4400_0000–0x4FFF_FFFF	Decorated AND, OR, XOR, LAC1, LAS1 references to peripherals and GPIO based at either 0x4000_F000 or 0x400F_F000
0x5000_0000–0x5FFF_FFFF	Decorated BFI, UBFX references to peripherals and GPIO only based at 0x4000_F000

## 39.4 Application information

In this section, GNU assembler macros with C expression operands are presented as examples of the required instructions to perform decorated operations.

This section specifically presents a partial bme.h file defining the assembly language expressions for decorated logical stores: AND, OR, and XOR. Comparable functions for BFI and the decorated loads are more complex and available in the complete BME header file.

These macros use the same function names presented in [Functional description](#).

```
#define IOANDW(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"      \
          "mov    r2, %[wdata];"     \
          "str    r2, [r3];"         \
          ":: [addr] \"r\" (ADDR), [wdata] \"r\" (WDATA) : \"r2\", \"r3\");

#define IOANDH(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"      \
          "mov    r2, %[wdata];"     \
          "strh   r2, [r3];"         \
          ":: [addr] \"r\" (ADDR), [wdata] \"r\" (WDATA) : \"r2\", \"r3\");

#define IOANDB(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"      \
          "mov    r2, %[wdata];"     \
          "strb   r2, [r3];"         \
          ":: [addr] \"r\" (ADDR), [wdata] \"r\" (WDATA) : \"r2\", \"r3\");
```

```

#define IOORW(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<27);"  \
          "orr    r3, %[addr];"    \
          "mov    r2, %[wdata];"   \
          "str    r2, [r3];"       \
          ":: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOORH(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<27);"  \
          "orr    r3, %[addr];"    \
          "mov    r2, %[wdata];"   \
          "strh   r2, [r3];"       \
          ":: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOORB(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<27);"  \
          "orr    r3, %[addr];"    \
          "mov    r2, %[wdata];"   \
          "strb   r2, [r3];"       \
          ":: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORW(ADDR,WDATA)         \
    __asm("ldr    r3, =(3<<26);"  \
          "orr    r3, %[addr];"    \
          "mov    r2, %[wdata];"   \
          "str    r2, [r3];"       \
          ":: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORH(ADDR,WDATA)         \
    __asm("ldr    r3, =(3<<26);"  \
          "orr    r3, %[addr];"    \
          "mov    r2, %[wdata];"   \
          "strh   r2, [r3];"       \
          ":: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORB(ADDR,WDATA)         \
    __asm("ldr    r3, =(3<<26);"  \
          "orr    r3, %[addr];"    \
          "mov    r2, %[wdata];"   \
          "strb   r2, [r3];"       \
          ":: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

```





# Chapter 40

## Micro Trace Buffer (MTB)

### 40.1 Introduction

Microcontrollers using the Cortex-M0+ processor core include support for a CoreSight Micro Trace Buffer to provide program trace capabilities.

The proper name for this function is the CoreSight Micro Trace Buffer for the Cortex-M0+ Processor; in this document, it is simply abbreviated as the MTB.

The simple program trace function creates instruction address change-of-flow data packets in a user-defined region of the system RAM. Accordingly, the system RAM controller manages requests from two sources:

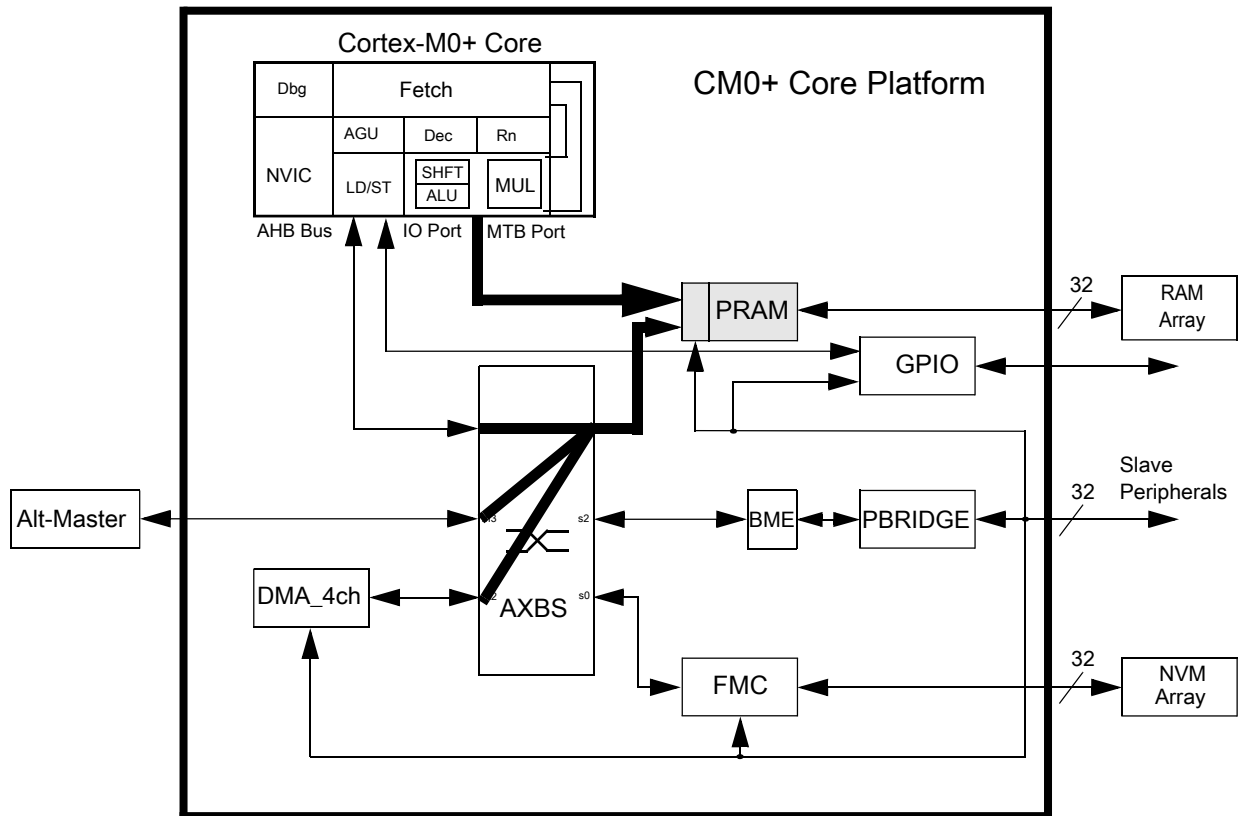
- AMBA-AHB reads and writes from the system bus
- program trace packet writes from the processor

As part of the MTB functionality, there is a DWT (Data Watchpoint and Trace) module that allows the user to define watchpoint addresses, or optionally, an address and data value, that when triggered, can be used to start or stop the program trace recording.

This document details the functionality of both the MTB\_RAM and MTB\_DWT capabilities.

#### 40.1.1 Overview

A generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers is shown as follows:



**Figure 40-1. Generic Cortex-M0+ core platform block diagram**

As shown in the block diagram, the platform RAM (PRAM) controller connects to two input buses:

- the crossbar slave port for system bus accesses
- a "private execution MTB port" from the core

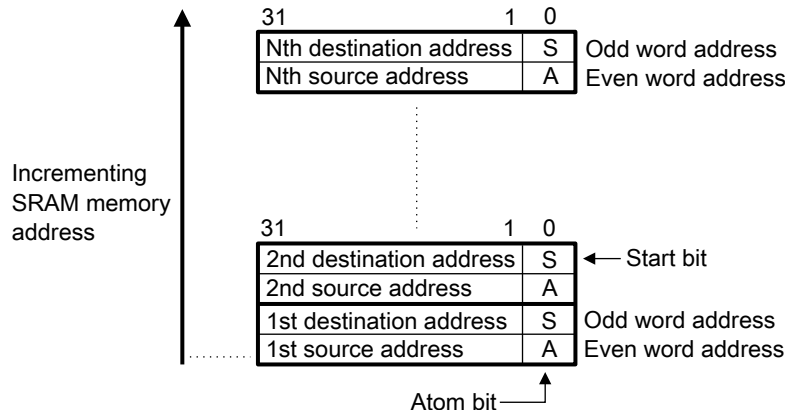
The logical paths from the crossbar master input ports to the PRAM controller are highlighted along with the private execution trace port from the processor core. The private MTB port signals the instruction address information needed for the 64-bit program trace packets written into the system RAM. The PRAM controller output interfaces to the attached RAM array. In this document, the PRAM controller is the MTB\_RAM controller.

The following information is taken from the ARM CoreSight Micro Trace Buffer documentation.

"The execution trace packet consists of a pair of 32-bit words that the MTB generates when it detects the processor PC value changes non-sequentially. A non-sequential PC change can occur during branch instructions or during exception entry.

The processor can cause a trace packet to be generated for any instruction.

The following figure shows how the execution trace information is stored in memory as a sequence of packets.



**Figure 40-2. MTB execution trace storage format**

The first, lower addressed, word contains the source of the branch, the address it branched from. The value stored only records bits[31:1] of the source address, because Thumb instructions are at least halfword aligned. The least significant bit of the value is the A-bit. The A-bit indicates the atomic state of the processor at the time of the branch, and can differentiate whether the branch originated from an instruction in a program, an exception, or a PC update in Debug state. When it is zero the branch originated from an instruction, when it is one the branch originated from an exception or PC update in Debug state. This word is always stored at an even word location.

The second, higher addressed word contains the destination of the branch, the address it branched to. The value stored only records bits[31:1] of the branch address. The least significant bit of the value is the S-bit. The S-bit indicates where the trace started. An S-bit value of 1 indicates where the first packet after the trace started and a value of 0 is used for other packets. Because it is possible to start and stop tracing multiple times in a trace session, the memory might contain several packets with the S-bit set to 1. This word is always stored in the next higher word in memory, an odd word address.

When the A-bit is set to 1, the source address field contains the architecturally-preferred return address for the exception. For example, if an exception was caused by an SVC instruction, then the source address field contains the address of the following instruction. This is different from the case where the A-bit is set to 0. In this case, the source address contains the address of the branch instruction.

For an exception return operation, two packets are generated:

- The first packet has the:
  - Source address field set to the address of the instruction that causes the exception return, BX or POP.

- Destination address field set to bits[31:1] of the EXC\_RETURN value. See the ARM v6-M Architecture Reference Manual.
- The A-bit set to 0.
- The second packet has the:
  - Source address field set to bits[31:1] of the EXC\_RETURN value.
  - Destination address field set to the address of the instruction where execution commences.
  - A-bit set to 1."

Given the recorded change-of-flow trace packets in system RAM and the memory image of the application, a debugger can read out the data and create an instruction-by-instruction program trace. In keeping with the low area and power implementation cost design targets, the MTB trace format is less efficient than other CoreSight trace modules, for example, the ETM (Embedded Trace Macrocell). Since each branch packet is 8 bytes in size, a 1 KB block of system RAM can contain 128 branches. Using the Dhrystone 2.1 benchmark's dynamic runtime as an example, this corresponds to about 875 instructions per KB of trace RAM, or with a zero wait state memory, this corresponds to approximately 1600 processor cycles per KB. This metric is obviously very sensitive to the runtime characteristics of the user code.

The MTB\_DWT function (not shown in the core platform block diagram) monitors the processor address and data buses so that configurable watchpoints can be detected to trigger the appropriate response in the MTB recording.

## 40.1.2 Features

The key features of the MTB\_RAM and MTB\_DWT include:

- Memory controller for system RAM and Micro Trace Buffer for program trace packets
- Read/write capabilities for system RAM accesses, write-only for program trace packets
- Supports zero wait state response to system bus accesses when no trace data is being written
- Can buffer two AHB address phases and one data write for system RAM accesses
- Supports 64-bit program trace packets including source and destination instruction addresses
- Program trace information in RAM available to MCU's application code or external debugger
- Program trace watchpoint configuration accessible by MCU's application code or debugger
- Location and size of RAM trace buffer is configured by software

- Two DWT comparators (addresses or address + data) provide programmable start/stop recording
- CoreSight compliant debug functionality

### 40.1.3 Modes of operation

The MTB\_RAM and MTB\_DWT functions do not support any special modes of operation. The MTB\_RAM controller, as a memory-mapped device located on the platform's slave AHB system bus, responds strictly on the basis of memory addresses for accesses to its attached RAM array. The MTB private execution bus provides program trace packet write information to the RAM controller. Both the MTB\_RAM and MTB\_DWT modules are memory-mapped, so their programming models can be accessed.

All functionality associated with the MTB\_RAM and MTB\_DWT modules resides in the core platform's clock domain; this includes its connections with the RAM array.

## 40.2 External signal description

The MTB\_RAM and MTB\_DWT modules do not directly support any external interfaces.

The internal interface includes a standard AHB bus with a 32-bit datapath width from the appropriate crossbar slave port plus the private execution trace bus from the processor core. The signals in the private execution trace bus are detailed in the following table taken from the ARM CoreSight Micro Trace Buffer documentation. The signal direction is defined as viewed by the MTB\_RAM controller.

**Table 40-1. Private execution trace port from the core to MTB\_RAM**

Signal	Direction	Description
LOCKUP	Input	Indicates the processor is in the Lockup state. This signal is driven LOW for cycles when the processor is executing normally and driven HIGH for every cycle the processor is waiting in the Lockup state. This signal is valid on every cycle.
IAESEQ	Input	Indicates the next instruction address in execute, IAEX, is sequential, that is non-branching.
IAEXEN	Input	IAEX register enable.
IAEX[30:0]	Input	Registered address of the instruction in the execution stage, shifted right by one bit, that is, PC >> 1.
ATOMIC	Input	Indicates the processor is performing non-instruction related activities.
EDBGREQ	Output	Request for the processor to enter the Debug state, if enabled, and halt.

In addition, there are two signals formed by the MTB\_DWT module and driven to the MTB\_RAM controller: TSTART (trace start) and TSTOP (trace stop). These signals can be configured using the trace watchpoints to define programmable addresses and data values to affect the program trace recording state.

## 40.3 Memory map and register definition

The MTB\_RAM and MTB\_DWT modules each support a sparsely-populated 4 KB address space for their programming models. For each address space, there are a variety of control and configurable registers near the base address, followed by a large unused address space and finally a set of CoreSight registers to support dynamic determination of the debug configuration for the device.

Accesses to the programming model follow standard ARM conventions. Taken from the ARM CoreSight Micro Trace Buffer documentation, these are:

- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in UNPREDICTABLE behavior.
- The behavior of the MTB is UNPREDICTABLE if the registers with UNKNOWN reset values are not programmed prior to enabling trace.
- Unless otherwise stated in the accompanying text:
  - Do not modify reserved register bits
  - Ignore reserved register bits on reads
  - All register bits are reset to a logic 0 by a system or power-on reset
  - Use only word size, 32-bit, transactions to access all registers

### 40.3.1 MTB\_RAM Memory Map

MTB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
F000_0000	MTB Position Register (MTB_POSITION)	32	R/W	Undefined	<a href="#">40.3.1.1/776</a>
F000_0004	MTB Master Register (MTB_MASTER)	32	R/W	<a href="#">See section</a>	<a href="#">40.3.1.2/777</a>
F000_0008	MTB Flow Register (MTB_FLOW)	32	R/W	Undefined	<a href="#">40.3.1.3/779</a>
F000_000C	MTB Base Register (MTB_BASE)	32	R	Undefined	<a href="#">40.3.1.4/781</a>
F000_0F00	Integration Mode Control Register (MTB_MODECTRL)	32	R	0000_0000h	<a href="#">40.3.1.5/781</a>

Table continues on the next page...

## MTB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
F000_0FA0	Claim TAG Set Register (MTB_TAGSET)	32	R	0000_0000h	40.3.1.6/ 782
F000_0FA4	Claim TAG Clear Register (MTB_TAGCLEAR)	32	R	0000_0000h	40.3.1.7/ 782
F000_0FB0	Lock Access Register (MTB_LOCKACCESS)	32	R	0000_0000h	40.3.1.8/ 783
F000_0FB4	Lock Status Register (MTB_LOCKSTAT)	32	R	0000_0000h	40.3.1.9/ 783
F000_0FB8	Authentication Status Register (MTB_AUTHSTAT)	32	R	0000_0000h	40.3.1.10/ 783
F000_0FBC	Device Architecture Register (MTB_DEVICEARCH)	32	R	4770_0A31h	40.3.1.11/ 784
F000_0FC8	Device Configuration Register (MTB_DEVICECFG)	32	R	0000_0000h	40.3.1.12/ 785
F000_0FCC	Device Type Identifier Register (MTB_DEVICETYPID)	32	R	0000_0031h	40.3.1.13/ 785
F000_0FD0	Peripheral ID Register (MTB_PERIPHID4)	32	R	See section	40.3.1.14/ 786
F000_0FD4	Peripheral ID Register (MTB_PERIPHID5)	32	R	See section	40.3.1.14/ 786
F000_0FD8	Peripheral ID Register (MTB_PERIPHID6)	32	R	See section	40.3.1.14/ 786
F000_0FDC	Peripheral ID Register (MTB_PERIPHID7)	32	R	See section	40.3.1.14/ 786
F000_0FE0	Peripheral ID Register (MTB_PERIPHID0)	32	R	See section	40.3.1.14/ 786
F000_0FE4	Peripheral ID Register (MTB_PERIPHID1)	32	R	See section	40.3.1.14/ 786
F000_0FE8	Peripheral ID Register (MTB_PERIPHID2)	32	R	See section	40.3.1.14/ 786
F000_0FEC	Peripheral ID Register (MTB_PERIPHID3)	32	R	See section	40.3.1.14/ 786
F000_0FF0	Component ID Register (MTB_COMPID0)	32	R	See section	40.3.1.15/ 786
F000_0FF4	Component ID Register (MTB_COMPID1)	32	R	See section	40.3.1.15/ 786
F000_0FF8	Component ID Register (MTB_COMPID2)	32	R	See section	40.3.1.15/ 786
F000_0FFC	Component ID Register (MTB_COMPID3)	32	R	See section	40.3.1.15/ 786

### 40.3.1.1 MTB Position Register (MTB\_POSITION)

The MTB\_POSITION register contains the Trace Write Address Pointer and Wrap fields. This register can be modified by the explicit programming model writes. It is also automatically updated by the MTB hardware when trace packets are being recorded.

The base address of the system RAM in the memory map dictates special consideration for the placement of the MTB. Consider the following guidelines:

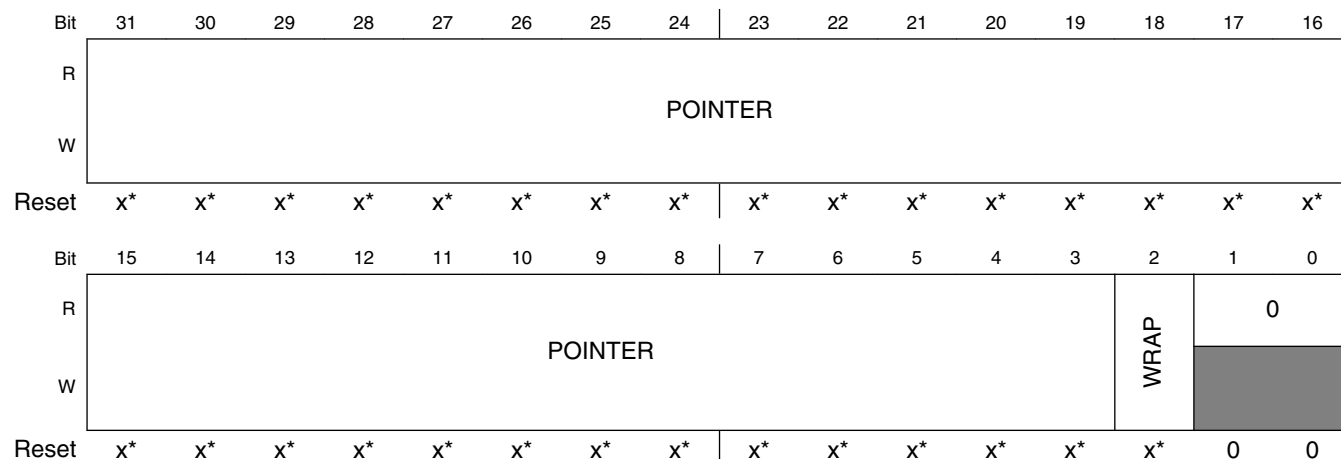
For the standard configuration where the size of the MTB is  $\leq 25\%$  of the total RAM capacity, it is recommended the MTB be based at the address defined by the MTB\_BASE register. The read-only MTB\_BASE register is defined by the expression  $(0x2000\_0000 - (RAM\_Size/4))$ . For this configuration, the MTB\_POSITION register is initialized to  $MTB\_BASE \& 0x0000\_7FF8$ .

If the size of the MTB is more than 25% but less than or equal to 50% of the total RAM capacity, it is recommended the MTB be based at address 0x2000\_0000. In this configuration, the MTB\_POSITION register is initialized to  $(0x2000\_0000 \& 0x0000\_7FF8) = 0x0000\_00000$ .

Following these two suggested placements provides a full-featured circular memory buffer containing program trace packets.

In the unlikely event an even larger trace buffer is required, a write-once capacity of 75% of the total RAM capacity can be based at address 0x2000\_0000. The MTB\_POSITION register is initialized to  $(0x2000\_0000 \& 0x0000\_7FF8) = 0x0000\_0000$ . However, this configuration cannot support operation as a circular queue and instead requires the use of the MTB\_FLOW[WATERMARK] capability to automatically disable tracing or halting the processor as the number of packet writes approach the buffer capacity. See the MTB\_FLOW register description for more details.

Address: F000\_0000h base + 0h offset = F000\_0000h





\* Notes:

- x = Undefined at reset.

### MTB\_POSITION field descriptions

Field	Description
31–3 POINTER	<p>Trace Packet Address Pointer[28:0]</p> <p>Because a packet consists of two words, the POINTER field is the address of the first word of a packet. This field contains bits[31:3] of the RAM address where the next trace packet is written. Therefore, it points to an unused location and is automatically incremented.</p> <p>A debug agent can calculate the system memory map address for the current location in the MTB using the following "generic" equation:</p> <p>Given <math>mtb\_size = 1 \ll (MTB\_MASTER[MASK] + 4)</math>,</p> <p><math>systemAddress = MTB\_BASE + (((MTB\_POSITION \&amp; 0xFFFF\_FFF8) + (mtb\_size - (MTB\_BASE \&amp; (mtb\_size-1)))) \&amp; 0x0000\_7FF8)</math>;</p> <p>For this device, a simpler expression also applies. See the following pseudo-code:</p> <p>if <math>((MTB\_POSITION \gg 13) == 0x3)</math> <math>systemAddress = (0x1FFF \ll 16) + (0x1 \ll 15) + (MTB\_POSITION \&amp; 0x7FF8)</math>; else <math>systemAddress = (0x2000 \ll 16) + (0x0 \ll 15) + (MTB\_POSITION \&amp; 0x7FF8)</math>;</p> <p><b>NOTE:</b> The size of the RAM is parameterized and the most significant bits of the POINTER field are RAZ/WI.</p> <p>For these devices, <math>POSITION[31:15] == POSITION[POINTER[28:12]]</math> are RAZ/WI. Therefore, the active bits in this field are <math>POSITION[14:3] == POSITION[POINTER[11:0]]</math>.</p>
2 WRAP	<p>WRAP</p> <p>This field is set to 1 automatically when the POINTER value wraps as determined by the MTB_MASTER[MASK] field in the MASTER Trace Control Register. A debug agent might use the WRAP field to determine whether the trace information above and below the pointer address is valid.</p>
Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>

#### 40.3.1.2 MTB Master Register (MTB\_MASTER)

The MTB\_MASTER register contains the main program trace enable plus other trace controls. This register can be modified by the explicit programming model writes. MTB\_MASTER[EN] and MTB\_MASTER[HALTREQ] fields are also automatically updated by the MTB hardware.

Before MTB\_MASTER[EN] or MTB\_MASTER[TSTARTEN] are set to 1, the software must initialize the MTB\_POSITION and MTB\_FLOW registers.

If MTB\_FLOW[WATERMARK] is used to stop tracing or to halt the processor, MTB\_MASTER[MASK] must still be set to a value that prevents MTB\_POSITION[POINTER] from wrapping before it reaches the MTB\_FLOW[WATERMARK] value.

**NOTE**

The format of this mask field is different than MTBDWT\_MASKn[MASK].

Address: F000\_0000h base + 4h offset = F000\_0004h

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						HALTREQ	RAMPRIV	SFRWPRIV	TSTOPEN	TSTARTEN	MASK				
W	[Shaded]						HALTREQ	RAMPRIV	SFRWPRIV	TSTOPEN	TSTARTEN	MASK				
Reset	0	0	0	0	0	0	0	0	1	0	0	x*	x*	x*	x*	x*

- \* Notes:
- x = Undefined at reset.

**MTB\_MASTER field descriptions**

Field	Description
31 EN	<p>Main Trace Enable</p> <p>When this field is 1, trace data is written into the RAM memory location addressed by MTB_POSITION[POINTER]. The MTB_POSITION[POINTER] value auto increments after the trace data packet is written.</p> <p>EN can be automatically set to 0 using the MTB_FLOW[WATERMARK] field and the MTB_FLOW[AUTOSTOP] bit.</p> <p>EN is automatically set to 1 if TSTARTEN is 1 and the TSTART signal is HIGH.</p> <p>EN is automatically set to 0 if TSTOPEN is 1 and the TSTOP signal is HIGH.</p> <p><b>NOTE:</b> If EN is set to 0 because MTB_FLOW[WATERMARK] is set, then it is not automatically set to 1 if TSTARTEN is 1 and the TSTART input is HIGH. In this case, tracing can only be restarted if MTB_FLOW[WATERMARK] or MTB_POSITION[POINTER] value is changed by software.</p>
30–10 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
9 HALTREQ	<p>Halt Request</p> <p>This field is connected to the halt request signal of the trace logic, EDBGREQ. When HALTREQ is set to 1, the EDBFGRQ is asserted if DBGEN (invasive debug enable, one of the debug authentication interface signals) is also HIGH. HALTREQ can be automatically set to 1 using MTB_FLOW[WATERMARK].</p>
8 RAMPRIV	<p>RAM Privilege</p> <p>If this field is 0, then user or privileged AHB read and write accesses to the RAM are permitted. If this field is 1, then only privileged AHB read and write accesses to the RAM are permitted and user accesses are RAZ/WI. The HPROT[1] signal determines if an access is a user or privileged mode reference.</p>

Table continues on the next page...

### MTB\_MASTER field descriptions (continued)

Field	Description
7 SFRWPRIV	<p>Special Function Register Write Privilege</p> <p>If this field is 0, then user or privileged AHB read and write accesses to the MTB_RAM Special Function Registers (programming model) are permitted. If this field is 1, then only privileged write accesses are permitted; user write accesses are ignored. The HPROT[1] signal determines if an access is user or privileged. Note MTB_RAM SFR read access are not controlled by this bit and are always permitted.</p>
6 TSTOPEN	<p>Trace Stop Input Enable</p> <p>If this field is 1 and the TSTOP signal is HIGH, then EN is set to 0. If a trace packet is being written to memory, the write is completed before tracing is stopped.</p>
5 TSTARTEN	<p>Trace Start Input Enable</p> <p>If this field is 1 and the TSTART signal is HIGH, then EN is set to 1. Tracing continues until a stop condition occurs.</p>
MASK	<p>Mask</p> <p>This value determines the maximum size of the trace buffer in RAM. It specifies the most-significant bit of the MTB_POSITION[POINTER] field that can be updated by automatic increment. If the trace tries to advance past this power of 2, the MTB_POSITION[WRAP] bit is set to 1, the MTB_POSITION[MASK+3:3] == MTB_POSITION[POINTER[MASK:0]] bits are set to 0, and the MTB_POSITION[14:MASK+3] == MTB_POSITION[POINTER[11:MASK+1]] bits remain unchanged.</p> <p>This field causes the trace packet information to be stored in a circular buffer of size <math>2^{[MASK+4]}</math> bytes, that can be positioned in memory at multiples of this size. As detailed in the MTB_POSITION description, typical "upper limits" for the MTB size are RAM_Size/4 or RAM_Size/2. Values greater than the maximum have the same effect as the maximum.</p>

#### 40.3.1.3 MTB Flow Register (MTB\_FLOW)

The MTB\_FLOW register contains the watermark address and the autostop/autohalt control bits.

If tracing is stopped using the watermark autostop feature, it cannot be restarted until software clears the watermark autostop. This can be achieved in one of the following ways:

- Changing the MTB\_POSITION[POINTER] field value to point to the beginning of the trace buffer, or
- Setting MTB\_FLOW[AUTOSTOP] = 0.

A debug agent can use MTB\_FLOW[AUTOSTOP] to fill the trace buffer once only without halting the processor.

A debug agent can use MTB\_FLOW[AUTOHALT] to fill the trace buffer once before causing the Cortex-M0+ processor to enter the Debug state. To enter Debug state, the Cortex-M0+ processor might have to perform additional branch type operations. Therefore, the MTB\_FLOW[WATERMARK] field must be set below the final entry in the trace buffer region.

## Memory map and register definition

Address: F000\_0000h base + 8h offset = F000\_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	WATERMARK															
W																
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	WATERMARK													0	AUTOHALT	AUTOSTOP
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	0	x*	x*

\* Notes:

- x = Undefined at reset.

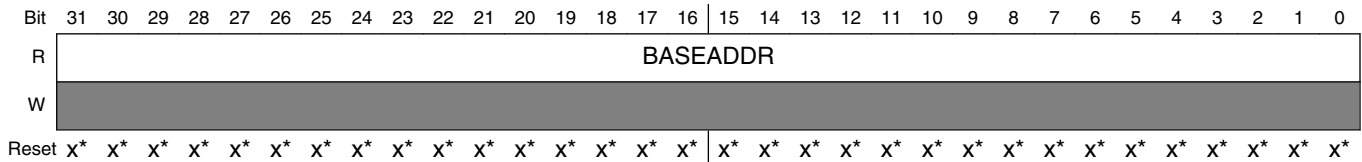
## MTB\_FLOW field descriptions

Field	Description
31–3 WATERMARK	<p>WATERMARK[28:0]</p> <p>This field contains an address in the same format as the MTB_POSITION[POINTER] field. When MTB_POSITION[POINTER] matches the WATERMARK field value, actions defined by the AUTOHALT and AUTOSTOP bits are performed.</p>
2 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
1 AUTOHALT	<p>AUTOHALT</p> <p>If this field is 1 and WATERMARK is equal to MTB_POSITION[POINTER], then MTB_MASTER[HALTREQ] is automatically set to 1. If the DBGGEN signal is HIGH, the MTB asserts this halt request to the Cortex-M0+ processor by asserting the EDBGREQ signal.</p>
0 AUTOSTOP	<p>AUTOSTOP</p> <p>If this field is 1 and WATERMARK is equal to MTB_POSITION[POINTER], then MTB_MASTER[EN] is automatically set to 0. This stops tracing.</p>

### 40.3.1.4 MTB Base Register (MTB\_BASE)

The read-only MTB\_BASE Register indicates where the RAM is located in the system memory map. This register is provided to enable auto discovery of the MTB RAM location, by a debug agent and is defined by a hardware design parameter. For this device, the base address is defined by the expression:  $MTB\_BASE[BASEADDR] = 0x2000\_0000 - (RAM\_Size/4)$

Address:  $F000\_0000h$  base + Ch offset =  $F000\_000Ch$



\* Notes:

- x = Undefined at reset.

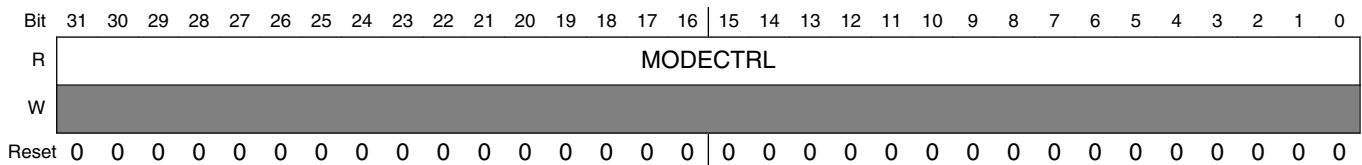
#### MTB\_BASE field descriptions

Field	Description
BASEADDR	BASEADDR  This value is defined with a hardwired signal and the expression: $0x2000\_0000 - (RAM\_Size/4)$ . For example, if the total RAM capacity is 16 KB, this field is $0x1FFF\_F000$ .

### 40.3.1.5 Integration Mode Control Register (MTB\_MODECTRL)

This register enables the device to switch from a functional mode, or default behavior, into integration mode. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address:  $F000\_0000h$  base + F00h offset =  $F000\_0F00h$



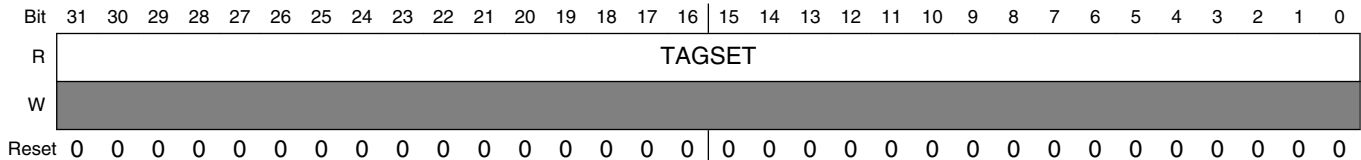
#### MTB\_MODECTRL field descriptions

Field	Description
MODECTRL	MODECTRL  Hardwired to $0x0000\_0000$

### 40.3.1.6 Claim TAG Set Register (MTB\_TAGSET)

The Claim Tag Set Register returns the number of bits that can be set on a read, and enables individual bits to be set on a write. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_0000h base + FA0h offset = F000\_0FA0h



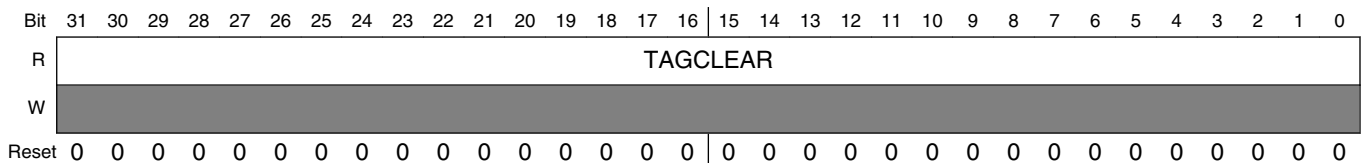
**MTB\_TAGSET field descriptions**

Field	Description
TAGSET	TAGSET Hardwired to 0x0000_0000

### 40.3.1.7 Claim TAG Clear Register (MTB\_TAGCLEAR)

The read/write Claim Tag Clear Register is used to read the claim status on debug resources. A read indicates the claim tag status. Writing 1 to a specific bit clears the corresponding claim tag to 0. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_0000h base + FA4h offset = F000\_0FA4h



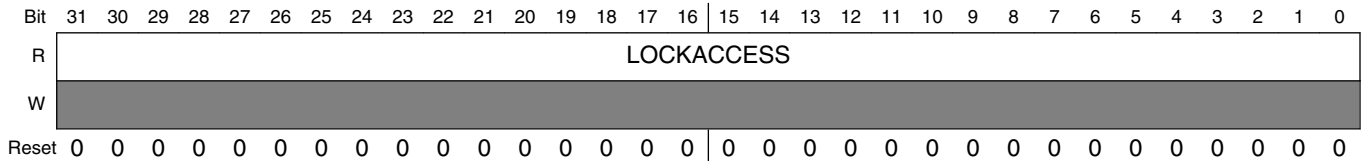
**MTB\_TAGCLEAR field descriptions**

Field	Description
TAGCLEAR	TAGCLEAR Hardwired to 0x0000_0000

### 40.3.1.8 Lock Access Register (MTB\_LOCKACCESS)

The Lock Access Register enables a write access to component registers. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_0000h base + FB0h offset = F000\_0FB0h



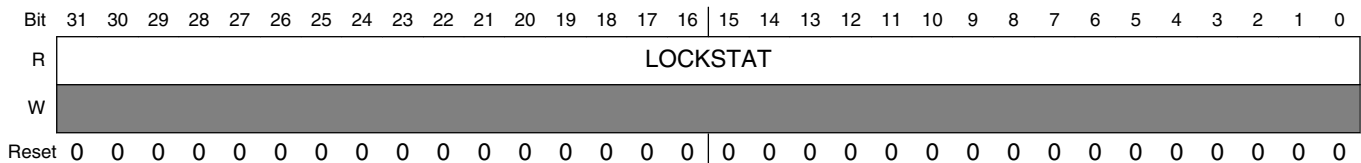
**MTB\_LOCKACCESS field descriptions**

Field	Description
LOCKACCESS	Hardwired to 0x0000_0000

### 40.3.1.9 Lock Status Register (MTB\_LOCKSTAT)

The Lock Status Register indicates the status of the lock control mechanism. This register is used in conjunction with the Lock Access Register. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_0000h base + FB4h offset = F000\_0FB4h



**MTB\_LOCKSTAT field descriptions**

Field	Description
LOCKSTAT	LOCKSTAT Hardwired to 0x0000_0000

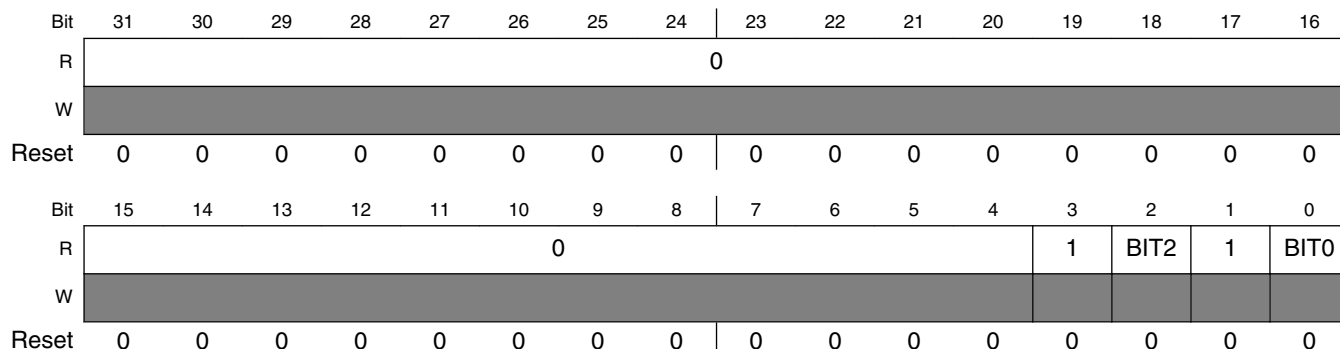
### 40.3.1.10 Authentication Status Register (MTB\_AUTHSTAT)

The Authentication Status Register reports the required security level and current status of the security enable bit pairs. Where functionality changes on a given security level, this change must be reported in this register. It is connected to specific signals used during the auto-discovery process by an external debug agent.

**Memory map and register definition**

MTB\_AUTHSTAT[3:2] indicates if nonsecure, noninvasive debug is enabled or disabled, while MTB\_AUTHSTAT[1:0] indicates the enabled/disabled state of nonsecure, invasive debug. For both 2-bit fields, 0b10 indicates the functionality is disabled and 0b11 indicates it is enabled.

Address: F000\_0000h base + FB8h offset = F000\_0FB8h



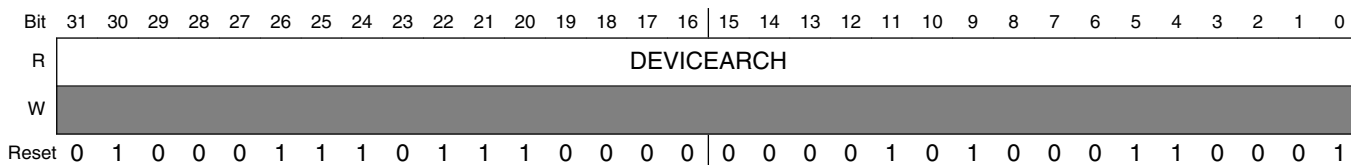
**MTB\_AUTHSTAT field descriptions**

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	BIT3 This read-only field is reserved and always has the value 1.
2 BIT2	BIT2 Connected to NIDEN or DBGEN signal.
1 Reserved	BIT1 This read-only field is reserved and always has the value 1.
0 BIT0	Connected to DBGEN.

**40.3.1.11 Device Architecture Register (MTB\_DEVICEARCH)**

This register indicates the device architecture. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_0000h base + FBCh offset = F000\_0FBCh





### MTB\_DEVICEARCH field descriptions

Field	Description
DEVICEARCH	DEVICEARCH Hardwired to 0x4770_0A31.

### 40.3.1.12 Device Configuration Register (MTB\_DEVICECFG)

This register indicates the device configuration. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_0000h base + FC8h offset = F000\_0FC8h

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

### MTB\_DEVICECFG field descriptions

Field	Description
DEVICECFG	DEVICECFG Hardwired to 0x0000_0000.

### 40.3.1.13 Device Type Identifier Register (MTB\_DEVICETYPID)

This register indicates the device type ID. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_0000h base + FCCh offset = F000\_0FCCh

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

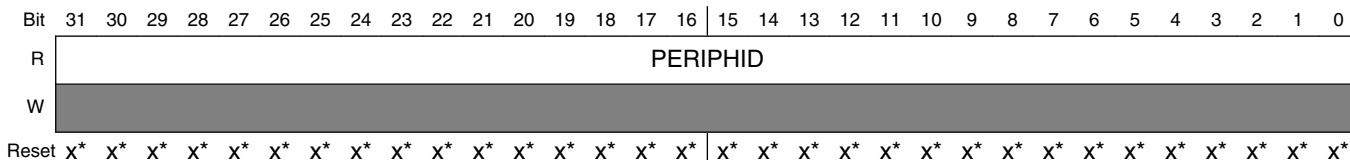
### MTB\_DEVICETYPID field descriptions

Field	Description
DEVICETYPID	DEVICETYPID Hardwired to 0x0000_0031.

### 40.3.1.14 Peripheral ID Register (MTB\_PERIPHIDn)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_0000h base + FD0h offset + (4d × i), where i=0d to 7d



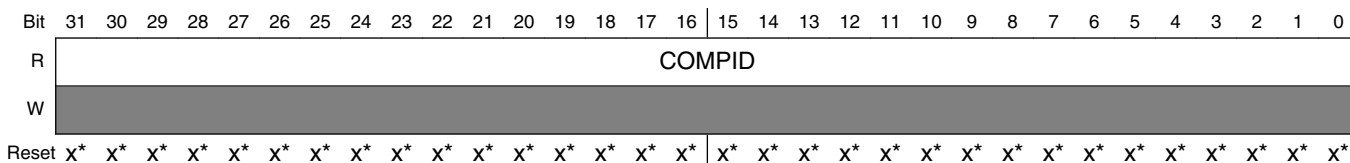
#### MTB\_PERIPHIDn field descriptions

Field	Description
PERIPHID	PERIPHID  Peripheral ID4 is hardwired to 0x0000_0004; ID0 to 0x0000_0032; ID1 to 0x0000_00B9; ID2 to 0x0000_001B; and all the others to 0x0000_0000.

### 40.3.1.15 Component ID Register (MTB\_COMPIDn)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_0000h base + FF0h offset + (4d × i), where i=0d to 3d



#### MTB\_COMPIDn field descriptions

Field	Description
COMPID	Component ID  Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0090; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.

## 40.3.2 MTB\_DWT Memory Map

The MTB\_DWT programming model supports a very simplified subset of the v7M debug architecture and follows the standard ARM DWT definition.

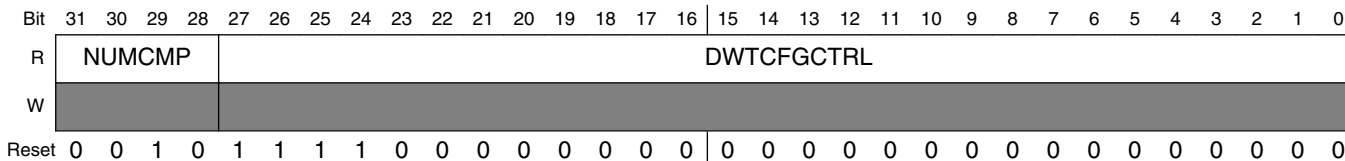
## MTBDWT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
F000_1000	MTB DWT Control Register (MTBDWT_CTRL)	32	R	2F00_0000h	40.3.2.1/ 788
F000_1020	MTB_DWT Comparator Register (MTBDWT_COMP0)	32	R/W	0000_0000h	40.3.2.2/ 789
F000_1024	MTB_DWT Comparator Mask Register (MTBDWT_MASK0)	32	R/W	0000_0000h	40.3.2.3/ 789
F000_1028	MTB_DWT Comparator Function Register 0 (MTBDWT_FCT0)	32	R/W	0000_0000h	40.3.2.4/ 790
F000_1030	MTB_DWT Comparator Register (MTBDWT_COMP1)	32	R/W	0000_0000h	40.3.2.2/ 789
F000_1034	MTB_DWT Comparator Mask Register (MTBDWT_MASK1)	32	R/W	0000_0000h	40.3.2.3/ 789
F000_1038	MTB_DWT Comparator Function Register 1 (MTBDWT_FCT1)	32	R/W	0000_0000h	40.3.2.5/ 792
F000_1200	MTB_DWT Trace Buffer Control Register (MTBDWT_TBCTRL)	32	R/W	2000_0000h	40.3.2.6/ 793
F000_1FC8	Device Configuration Register (MTBDWT_DEVICECFG)	32	R	0000_0000h	40.3.2.7/ 795
F000_1FCC	Device Type Identifier Register (MTBDWT_DEVICETYPEID)	32	R	0000_0004h	40.3.2.8/ 795
F000_1FD0	Peripheral ID Register (MTBDWT_PERIPHID4)	32	R	See section	40.3.2.9/ 796
F000_1FD4	Peripheral ID Register (MTBDWT_PERIPHID5)	32	R	See section	40.3.2.9/ 796
F000_1FD8	Peripheral ID Register (MTBDWT_PERIPHID6)	32	R	See section	40.3.2.9/ 796
F000_1FDC	Peripheral ID Register (MTBDWT_PERIPHID7)	32	R	See section	40.3.2.9/ 796
F000_1FE0	Peripheral ID Register (MTBDWT_PERIPHID0)	32	R	See section	40.3.2.9/ 796
F000_1FE4	Peripheral ID Register (MTBDWT_PERIPHID1)	32	R	See section	40.3.2.9/ 796
F000_1FE8	Peripheral ID Register (MTBDWT_PERIPHID2)	32	R	See section	40.3.2.9/ 796
F000_1FEC	Peripheral ID Register (MTBDWT_PERIPHID3)	32	R	See section	40.3.2.9/ 796
F000_1FF0	Component ID Register (MTBDWT_COMPID0)	32	R	See section	40.3.2.10/ 796
F000_1FF4	Component ID Register (MTBDWT_COMPID1)	32	R	See section	40.3.2.10/ 796
F000_1FF8	Component ID Register (MTBDWT_COMPID2)	32	R	See section	40.3.2.10/ 796
F000_1FFC	Component ID Register (MTBDWT_COMPID3)	32	R	See section	40.3.2.10/ 796

### 40.3.2.1 MTB DWT Control Register (MTBDWT\_CTRL)

The MTBDWT\_CTRL register provides read-only information on the watchpoint configuration for the MTB\_DWT.

Address: F000\_1000h base + 0h offset = F000\_1000h



#### MTBDWT\_CTRL field descriptions

Field	Description
31–28 NUMCMP	Number of comparators  The MTB_DWT implements two comparators.
DWTCFGCTRL	DWT configuration controls  This field is hardwired to 0xF00_0000, disabling all the remaining DWT functionality. The specific fields and their state are:  MTBDWT_CTRL[27] = NOTRCPKT = 1, trace sample and exception trace is not supported MTBDWT_CTRL[26] = NOEXTTRIG = 1, external match signals are not supported MTBDWT_CTRL[25] = NOCYCCNT = 1, cycle counter is not supported MTBDWT_CTRL[24] = NOPRFCNT = 1, profiling counters are not supported MTBDWT_CTRL[22] = CYCEBTENA = 0, no POSTCNT underflow packets generated MTBDWT_CTRL[21] = FOLDEVTENA = 0, no folded instruction counter overflow events MTBDWT_CTRL[20] = LSUEVTENA = 0, no LSU counter overflow events MTBDWT_CTRL[19] = SLEEPEVTENA = 0, no sleep counter overflow events MTBDWT_CTRL[18] = EXCEVTENA = 0, no exception overhead counter events MTBDWT_CTRL[17] = CPIEVTENA = 0, no CPI counter overflow events MTBDWT_CTRL[16] = EXCTRCENA = 0, generation of exception trace disabled MTBDWT_CTRL[12] = PCSAMPLENA = 0, no periodic PC sample packets generated MTBDWT_CTRL[11:10] = SYNCTAP = 0, no synchronization packets MTBDWT_CTRL[9] = CYCTAP = 0, cycle counter is not supported MTBDWT_CTRL[8:5] = POSTINIT = 0, cycle counter is not supported MTBDWT_CTRL[4:1] = POSTPRESET = 0, cycle counter is not supported MTBDWT_CTRL[0] = CYCCNTENA = 0, cycle counter is not supported

### 40.3.2.2 MTB\_DWT Comparator Register (MTBDWT\_COMPn)

The MTBDWT\_COMPn registers provide the reference value for comparator n.

Address: F000\_1000h base + 20h offset + (16d × i), where i=0d to 1d

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

#### MTBDWT\_COMPn field descriptions

Field	Description
COMP	Reference value for comparison  If MTBDWT_COMP0 is used for a data value comparator and the access size is byte or halfword, the data value must be replicated across all appropriate byte lanes of this register. For example, if the data is a byte-sized "x" value, then COMP[31:24] = COMP[23:16] = COMP[15:8] = COMP[7:0] = "x". Likewise, if the data is a halfword-size "y" value, then COMP[31:16] = COMP[15:0] = "y".

### 40.3.2.3 MTB\_DWT Comparator Mask Register (MTBDWT\_MASKn)

The MTBDWT\_MASKn registers define the size of the ignore mask applied to the reference address for address range matching by comparator n. Note the format of this mask field is different than the MTB\_MASTER[MASK].

Address: F000\_1000h base + 24h offset + (16d × i), where i=0d to 1d

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

#### MTBDWT\_MASKn field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MASK	MASK  The value of the ignore mask, 0-31 bits, is applied to address range matching. MASK = 0 is used to include all bits of the address in the comparison, except if MASK = 0 and the comparator is configured to watch instruction fetch addresses, address bit [0] is ignored by the hardware since all fetches must be at least halfword aligned. For MASK != 0 and regardless of watch type, address bits [x-1:0] are ignored in the address comparison.  Using a mask means the comparator matches on a range of addresses, defined by the unmasked most significant bits of the address, bits [31:x]. The maximum MASK value is 24, producing a 16 Mbyte mask. An attempted write of a MASK value > 24 is limited by the MTBDWT hardware to 24.

Table continues on the next page...

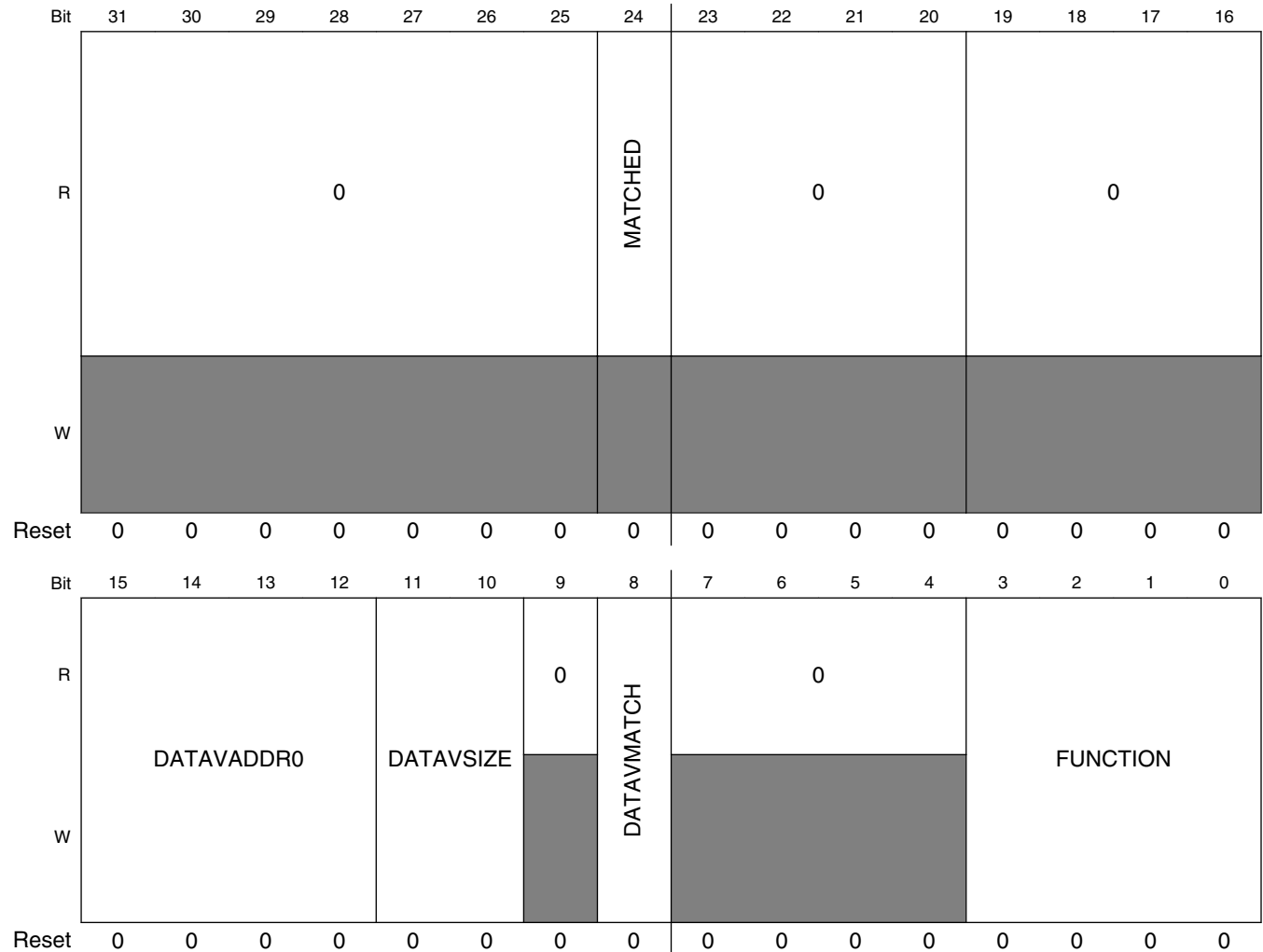
**MTBDWT\_MASKn field descriptions (continued)**

Field	Description
	If MTBDWT_COMP0 is used as a data value comparator, then MTBDWT_MASK0 should be programmed to zero.

**40.3.2.4 MTB\_DWT Comparator Function Register 0 (MTBDWT\_FCT0)**

The MTBDWT\_FCTn registers control the operation of comparator n.

Address: F000\_1000h base + 28h offset = F000\_1028h



**MTBDWT\_FCT0 field descriptions**

Field	Description
31-25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

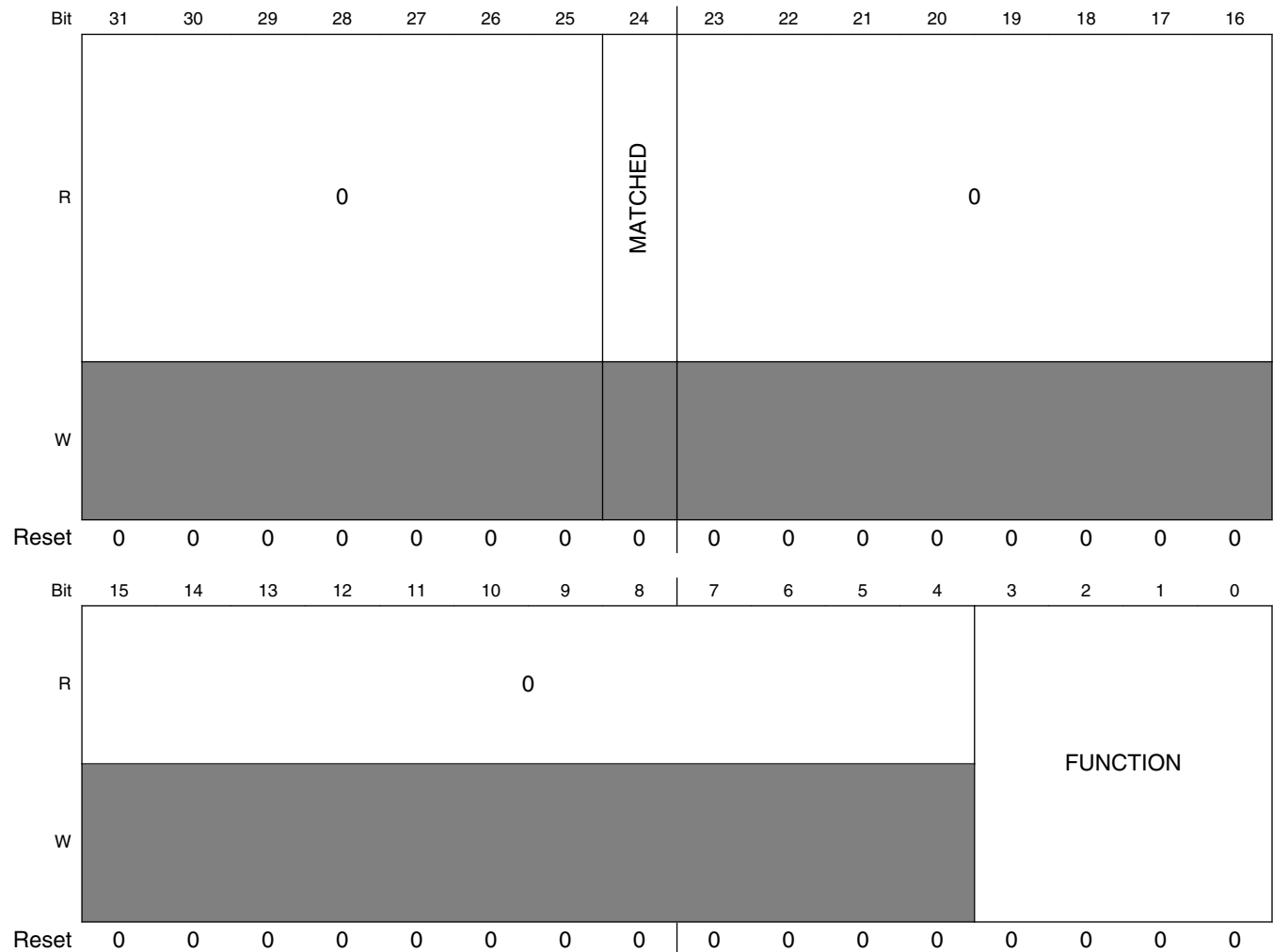
## MTBDWT\_FCT0 field descriptions (continued)

Field	Description
24 MATCHED	<p>Comparator match</p> <p>If this read-only flag is asserted, it indicates the operation defined by the FUNCTION field occurred since the last read of the register. Reading the register clears this bit.</p> <p>0 No match. 1 Match occurred.</p>
23–20 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
19–16 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
15–12 DATAVADDR0	<p>Data Value Address 0</p> <p>Since the MTB_DWT implements two comparators, the DATAVADDR0 field is restricted to values {0,1}. When the DATAVMATCH bit is asserted, this field defines the comparator number to use for linked address comparison.</p> <p>If MTBDWT_COMP0 is used as a data watchpoint and MTBDWT_COMP1 as an address watchpoint, DATAVADDR0 must be set.</p>
11–10 DATAVSIZE	<p>Data Value Size</p> <p>For data value matching, this field defines the size of the required data comparison.</p> <p>00 Byte. 01 Halfword. 10 Word. 11 Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.</p>
9 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
8 DATAVMATCH	<p>Data Value Match</p> <p>When this field is 1, it enables data value comparison. For this implementation, MTBDWT_COMP0 supports address or data value comparisons; MTBDWT_COMP1 only supports address comparisons.</p> <p>0 Perform address comparison. 1 Perform data value comparison.</p>
7–4 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>
FUNCTION	<p>Function</p> <p>Selects the action taken on a comparator match. If MTBDWT_COMP0 is used for a data value and MTBDWT_COMP1 for an address value, then MTBDWT_FCT1[FUNCTION] must be set to zero. For this configuration, MTBDWT_MASK1 can be set to a non-zero value, so the combined comparators match on a range of addresses.</p> <p>0000 Disabled. 0100 Instruction fetch. 0101 Data operand read. 0110 Data operand write. 0111 Data operand (read + write). others Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.</p>

### 40.3.2.5 MTB\_DWT Comparator Function Register 1 (MTBDWT\_FCT1)

The MTBDWT\_FCTn registers control the operation of comparator n. Since the MTB\_DWT only supports data value comparisons on comparator 0, there are several fields in the MTBDWT\_FCT1 register that are RAZ/WI (bits 12, 11:10, 8).

Address: F000\_1000h base + 38h offset = F000\_1038h



**MTBDWT\_FCT1 field descriptions**

Field	Description
31–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 MATCHED	Comparator match  If this read-only flag is asserted, it indicates the operation defined by the FUNCTION field occurred since the last read of the register. Reading the register clears this bit.

*Table continues on the next page...*



## MTBDWT\_FCT1 field descriptions (continued)

Field	Description
	0 No match. 1 Match occurred.
23–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
FUNCTION	Function  Selects the action taken on a comparator match. If MTBDWT_COMP0 is used for a data value and MTBDWT_COMP1 for an address value, then MTBDWT_FCT1[FUNCTION] must be set to zero. For this configuration, MTBDWT_MASK1 can be set to a non-zero value, so the combined comparators match on a range of addresses.  0000 Disabled. 0100 Instruction fetch. 0101 Data operand read. 0110 Data operand write. 0111 Data operand (read + write). others Reserved. Any attempts to use this value results in UNPREDICTABLE behavior.

### 40.3.2.6 MTB\_DWT Trace Buffer Control Register (MTBDWT\_TBCTRL)

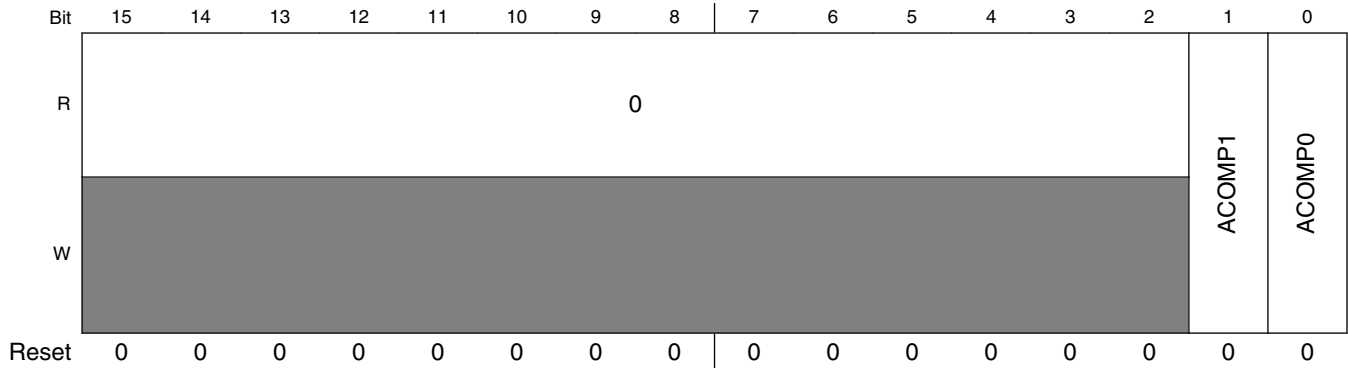
The MTBDWT\_TBCTRL register defines how the watchpoint comparisons control the actual trace buffer operation.

Recall the MTB supports starting and stopping the program trace based on the watchpoint comparisons signaled via TSTART and TSTOP. The watchpoint comparison signals are enabled in the MTB's control logic by setting the appropriate enable bits, MTB\_MASTER[TSTARTEN, TSTOPEN]. In the event of simultaneous assertion of both TSTART and TSTOP, TSTART takes priority.

Address: F000\_1000h base + 200h offset = F000\_1200h

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

## Memory map and register definition



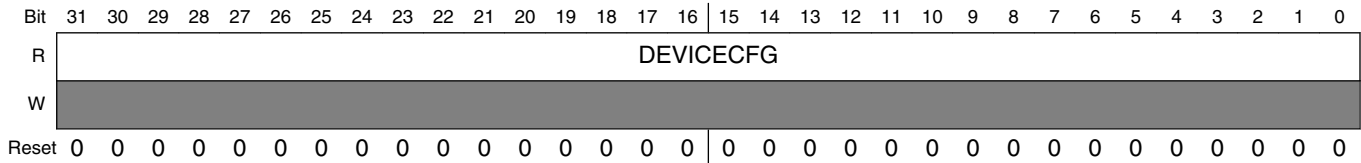
### MTBDWT\_TBCTRL field descriptions

Field	Description
31–28 NUMCOMP	<p>Number of Comparators</p> <p>This read-only field specifies the number of comparators in the MTB_DWT. This implementation includes two registers.</p>
27–2 Reserved	<p>This field is reserved.</p> <p>This read-only field is reserved and always has the value 0.</p>
1 ACOMP1	<p>Action based on Comparator 1 match</p> <p>When the MTBDWT_FCT1[MATCHED] is set, it indicates MTBDWT_COMP1 address compare has triggered and the trace buffer's recording state is changed.</p> <p>0 Trigger TSTOP based on the assertion of MTBDWT_FCT1[MATCHED]. 1 Trigger TSTART based on the assertion of MTBDWT_FCT1[MATCHED].</p>
0 ACOMP0	<p>Action based on Comparator 0 match</p> <p>When the MTBDWT_FCT0[MATCHED] is set, it indicates MTBDWT_COMP0 address compare has triggered and the trace buffer's recording state is changed. The assertion of MTBDWT_FCT0[MATCHED] is caused by the following conditions:</p> <ul style="list-style-type: none"> <li>Address match in MTBDWT_COMP0 when MTBDWT_FCT0[DATAVMATCH] = 0</li> <li>Data match in MTBDWT_COMP0 when MTBDWT_FCT0[DATAVMATCH, DATAVADDR0] = {1,0}</li> <li>Data match in MTBDWT_COMP0 and address match in MTBDWT_COMP1 when MTBDWT_FCT0[DATAVMATCH, DATAVADDR0] = {1,1}</li> </ul> <p>0 Trigger TSTOP based on the assertion of MTBDWT_FCT0[MATCHED]. 1 Trigger TSTART based on the assertion of MTBDWT_FCT0[MATCHED].</p>

### 40.3.2.7 Device Configuration Register (MTBDWT\_DEVICECFG)

This register indicates the device configuration. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_1000h base + FC8h offset = F000\_1FC8h



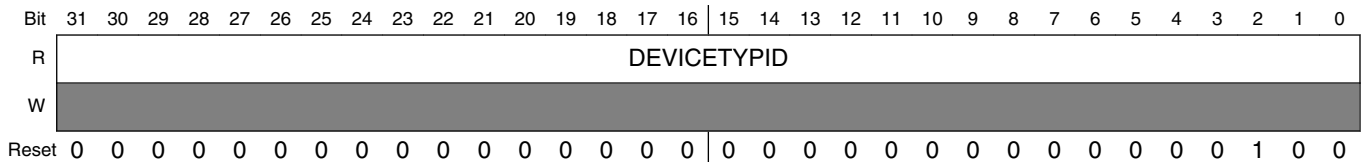
#### MTBDWT\_DEVICECFG field descriptions

Field	Description
DEVICECFG	DEVICECFG Hardwired to 0x0000_0000.

### 40.3.2.8 Device Type Identifier Register (MTBDWT\_DEVICETYPID)

This register indicates the device type ID. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_1000h base + FCCh offset = F000\_1FCCh



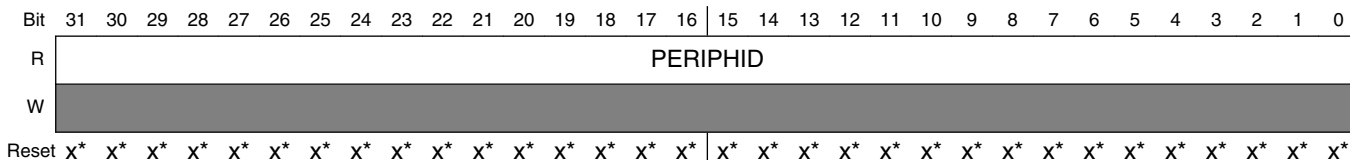
#### MTBDWT\_DEVICETYPID field descriptions

Field	Description
DEVICETYPID	DEVICETYPID Hardwired to 0x0000_0004.

### 40.3.2.9 Peripheral ID Register (MTBDWT\_PERIPHIDn)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_1000h base + FD0h offset + (4d × i), where i=0d to 7d



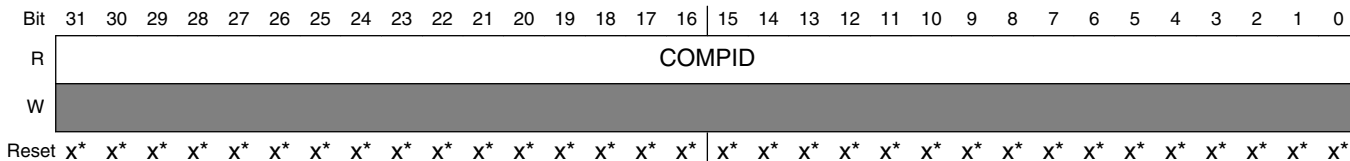
#### MTBDWT\_PERIPHIDn field descriptions

Field	Description
PERIPHID	PERIPHID Peripheral ID1 is hardwired to 0x0000_00E0; ID2 to 0x0000_0008; and all the others to 0x0000_0000.

### 40.3.2.10 Component ID Register (MTBDWT\_COMPIDn)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_1000h base + FF0h offset + (4d × i), where i=0d to 3d



#### MTBDWT\_COMPIDn field descriptions

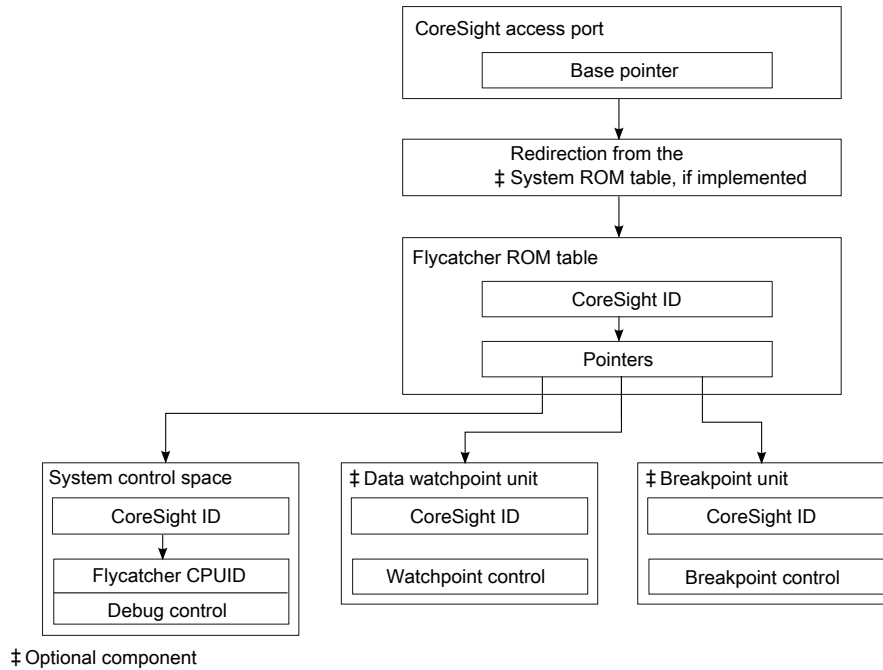
Field	Description
COMPID	Component ID Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0090; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.

## 40.3.3 System ROM Memory Map

The System ROM Table registers are also mapped into a sparsely-populated 4 KB address space.

For core configurations like that supported by Cortex-M0+, ARM recommends that a debugger identifies and connects to the debug components using the CoreSight debug infrastructure.

ARM recommends that a debugger follows the flow as shown in the following figure to discover the components in the CoreSight debug infrastructure. In this case, a debugger reads the peripheral and component ID registers for each CoreSight component in the CoreSight system.



**Figure 40-3. CoreSight discovery process**

**ROM memory map**

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
F000_2000	Entry (ROM_ENTRY0)	32	R	See section	40.3.3.1/798
F000_2004	Entry (ROM_ENTRY1)	32	R	See section	40.3.3.1/798
F000_2008	Entry (ROM_ENTRY2)	32	R	See section	40.3.3.1/798
F000_200C	End of Table Marker Register (ROM_TABLEMARK)	32	R	0000_0000h	40.3.3.2/799
F000_2FCC	System Access Register (ROM_SYSACCESS)	32	R	0000_0001h	40.3.3.3/799
F000_2FD0	Peripheral ID Register (ROM_PERIPHID4)	32	R	See section	40.3.3.4/800

Table continues on the next page...

**ROM memory map (continued)**

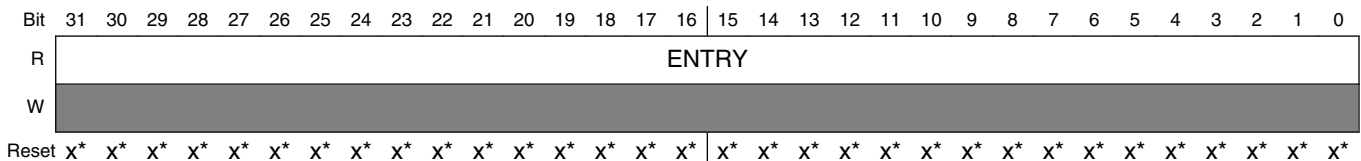
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
F000_2FD4	Peripheral ID Register (ROM_PERIPHID5)	32	R	See section	40.3.3.4/800
F000_2FD8	Peripheral ID Register (ROM_PERIPHID6)	32	R	See section	40.3.3.4/800
F000_2FDC	Peripheral ID Register (ROM_PERIPHID7)	32	R	See section	40.3.3.4/800
F000_2FE0	Peripheral ID Register (ROM_PERIPHID0)	32	R	See section	40.3.3.4/800
F000_2FE4	Peripheral ID Register (ROM_PERIPHID1)	32	R	See section	40.3.3.4/800
F000_2FE8	Peripheral ID Register (ROM_PERIPHID2)	32	R	See section	40.3.3.4/800
F000_2FEC	Peripheral ID Register (ROM_PERIPHID3)	32	R	See section	40.3.3.4/800
F000_2FF0	Component ID Register (ROM_COMPID0)	32	R	See section	40.3.3.5/800
F000_2FF4	Component ID Register (ROM_COMPID1)	32	R	See section	40.3.3.5/800
F000_2FF8	Component ID Register (ROM_COMPID2)	32	R	See section	40.3.3.5/800
F000_2FFC	Component ID Register (ROM_COMPID3)	32	R	See section	40.3.3.5/800

**40.3.3.1 Entry (ROM\_ENTRY<sub>n</sub>)**

The System ROM Table begins with "n" relative 32-bit addresses, one for each debug component present in the device and terminating with an all-zero value signaling the end of the table at the "n+1"-th value.

It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_2000h base + 0h offset + (4d × i), where i=0d to 2d



**ROM\_ENTRY<sub>n</sub> field descriptions**

Field	Description
ENTRY	ENTRY

**ROM\_ENTRY $n$  field descriptions (continued)**

Field	Description
	Entry 0 (MTB) is hardwired to 0xFFFF_E003; Entry 1 (MTBDWT) to 0xFFFF_F003; Entry 2 (CM0+ ROM Table) to 0xF00F_D003.

**40.3.3.2 End of Table Marker Register (ROM\_TABLEMARK)**

This register indicates end of table marker. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_2000h base + Ch offset = F000\_200Ch

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

**ROM\_TABLEMARK field descriptions**

Field	Description
MARK	MARK Hardwired to 0x0000_0000

**40.3.3.3 System Access Register (ROM\_SYSACCESS)**

This register indicates system access. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_2000h base + FCCh offset = F000\_2FCCh

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

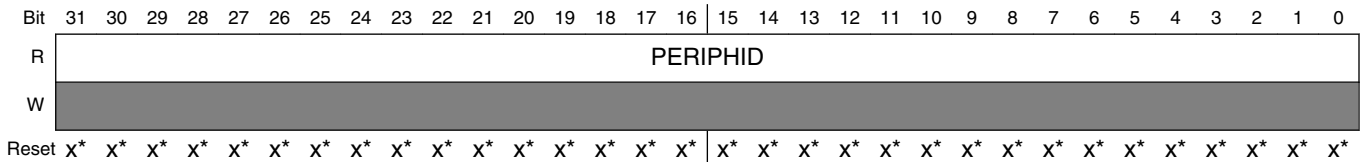
**ROM\_SYSACCESS field descriptions**

Field	Description
SYSACCESS	SYSACCESS Hardwired to 0x0000_0001

### 40.3.3.4 Peripheral ID Register (ROM\_PERIPHIDn)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_2000h base + FD0h offset + (4d × i), where i=0d to 7d



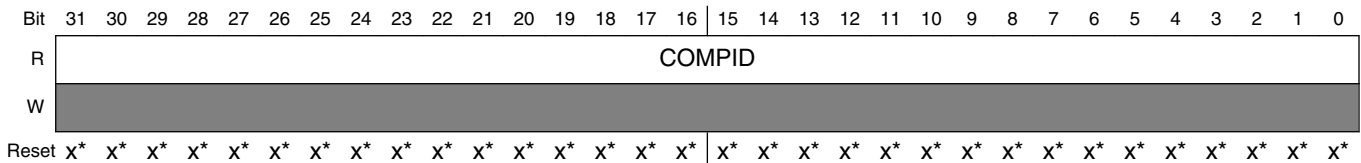
#### ROM\_PERIPHIDn field descriptions

Field	Description
PERIPHID	PERIPHID Peripheral ID1 is hardwired to 0x0000_00E0; ID2 to 0x0000_0008; and all the others to 0x0000_0000.

### 40.3.3.5 Component ID Register (ROM\_COMPIDn)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000\_2000h base + FF0h offset + (4d × i), where i=0d to 3d



#### ROM\_COMPIDn field descriptions

Field	Description
COMPID	Component ID Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0010; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.



# Chapter 41

## Cyclic Redundancy Check (CRC)

### 41.1 Introduction

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.

#### 41.1.1 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

#### 41.1.2 Block diagram

The following is a block diagram of the CRC.

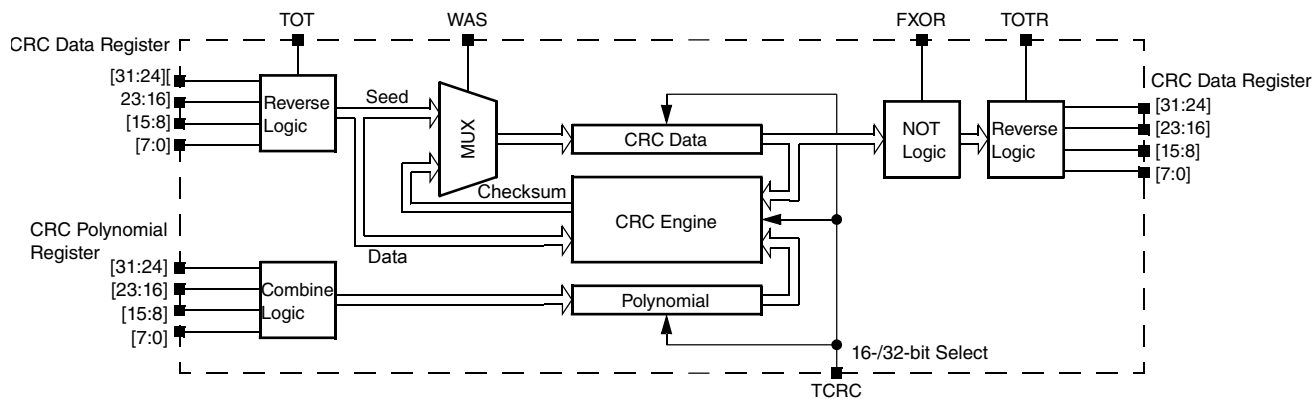


Figure 41-1. Programmable cyclic redundancy check (CRC) block diagram

### 41.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

#### 41.1.3.1 Run mode

This is the basic mode of operation.

#### 41.1.3.2 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.

## 41.2 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="#">41.2.1/803</a>
4003_2004	CRC Polynomial register (CRC_GPOLY)	32	R/W	0000_1021h	<a href="#">41.2.2/804</a>
4003_2008	CRC Control register (CRC_CTRL)	32	R/W	0000_0000h	<a href="#">41.2.3/804</a>

### 41.2.1 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	HU								HL								LU								LL							
W	1								1								1								1							
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

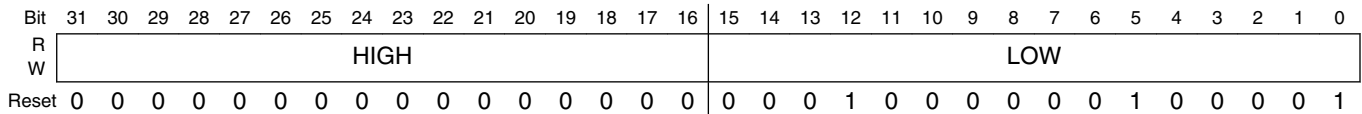
#### CRC\_DATA field descriptions

Field	Description
31–24 HU	CRC High Upper Byte  In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
23–16 HL	CRC High Lower Byte  In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
15–8 LU	CRC Low Upper Byte  When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.
LL	CRC Low Lower Byte  When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.

### 41.2.2 CRC Polynomial register (CRC\_GPOLY)

This register contains the value of the polynomial for the CRC calculation. The HIGH field contains the upper 16 bits of the CRC polynomial, which are used only in 32-bit CRC mode. Writes to the HIGH field are ignored in 16-bit CRC mode. The LOW field contains the lower 16 bits of the CRC polynomial, which are used in both 16- and 32-bit CRC modes.

Address: 4003\_2000h base + 4h offset = 4003\_2004h



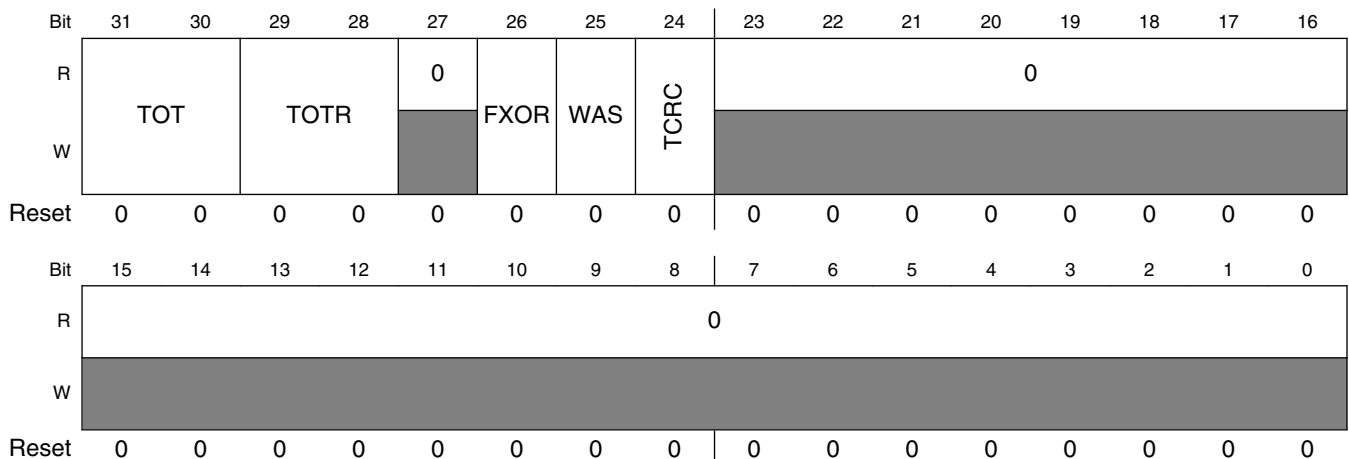
#### CRC\_GPOLY field descriptions

Field	Description
31–16 HIGH	High Polynominal Half-word  Writable and readable in 32-bit CRC mode (CTRL[TCRC] is 1). This field is not writable in 16-bit CRC mode (CTRL[TCRC] is 0).
LOW	Low Polynominal Half-word  Writable and readable in both 32-bit and 16-bit CRC modes.

### 41.2.3 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>
Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>

### 41.3 Functional description

### 41.3.1 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.

### 41.3.2 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.

#### 41.3.2.1 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.

### 41.3.2.2 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 byte-wise, 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.

### 41.3.3 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.

#### 41.3.3.1 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.

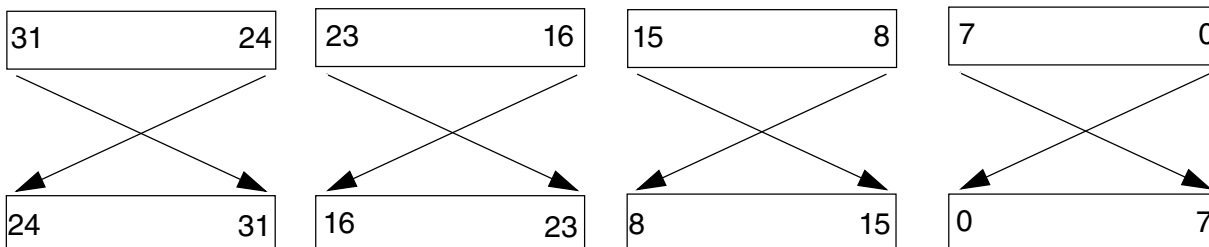
**Functional description**

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]}

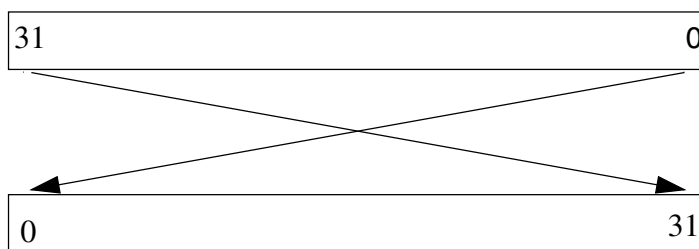


**Figure 41-2. Transpose type 01**

3. CTRL[TOT] or CTRL[TOTR] is 10.

Both bits in bytes and bytes are transposed.

reg[31:0] becomes = {reg[0:7], reg[8:15], reg[16:23], reg[24:31]}



**Figure 41-3. Transpose type 10**

4. CTRL[TOT] or CTRL[TOTR] is 11.

Bytes are transposed, but bits are not transposed.

reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}



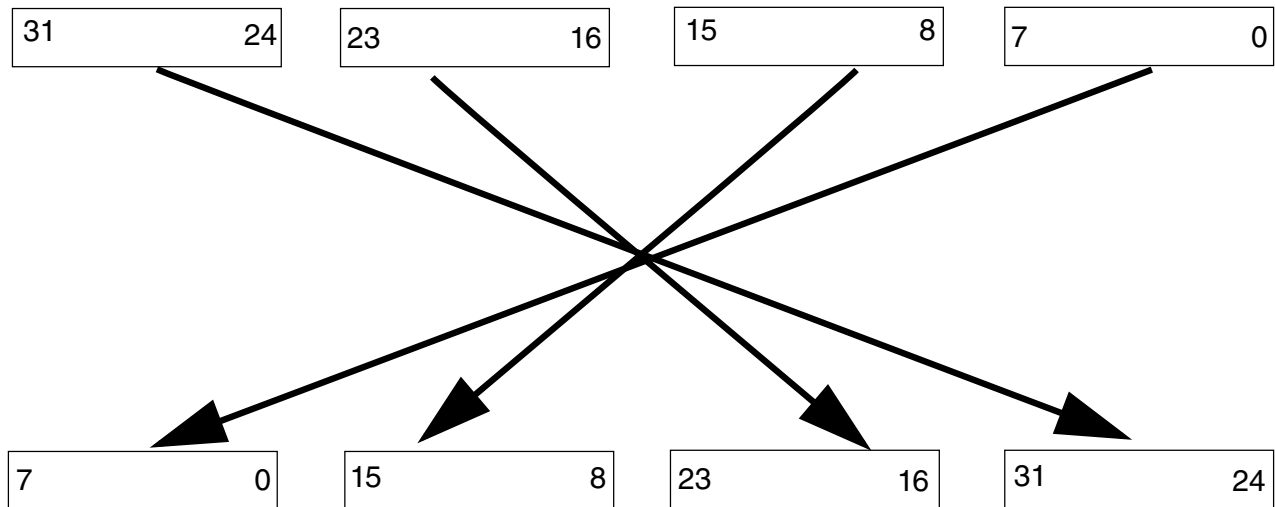


Figure 41-4. Transpose type 11

**NOTE**

For 8-bit and 16-bit write accesses to the CRC data register, the data is transposed with zeros on the unused byte or bytes (taking 32 bits as a whole), but the CRC is calculated on the valid byte(s) only. When reading the CRC data register for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in the CRC[**HU:HL**] fields. The user software must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

**41.3.4 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 42

## Flash Memory Controller (FMC)

### 42.1 Introduction

The Flash Memory Controller (FMC) is a memory acceleration unit. A list of features provided by the FMC can be found here.

- an interface between bus masters and the 32-bit program flash memory.
- a buffer and a cache that can accelerate program flash memory data transfers.

#### 42.1.1 Overview

The Flash Memory Controller manages the interface between bus masters and the 32-bit program flash memory. The FMC receives status information detailing the configuration of the flash memory and uses this information to ensure a proper interface. The FMC supports 8-bit, 16-bit, and 32-bit read operations from the program flash memory. A write operation to program flash memory results in a bus error.

In addition, the FMC provides two separate mechanisms for accelerating the interface between bus masters and program flash memory. A 32-bit speculation buffer can prefetch the next 32-bit flash memory location, and a 4-way, 4-set program flash memory cache can store previously accessed program flash memory data for quick access times.

#### 42.1.2 Features

The features of FMC module include:

- Interface between bus masters and the 32-bit program flash memory:
  - 8-bit, 16-bit, and 32-bit read operations to nonvolatile flash memory.
- Acceleration of data transfer from the program flash memory to the device:

- 32-bit prefetch speculation buffer for program flash accesses with controls for instruction/data access
- 4-way, 4-set, 32-bit line size program flash memory cache for a total of sixteen 32-bit entries with invalidation control

## 42.2 Modes of operation

The FMC operates only when a bus master accesses the program flash memory.

In terms of chip power modes:

- The FMC operates only in Run and Wait modes, including VLPR and VLPW modes.
- For any power mode where the program flash memory cannot be accessed, the FMC is disabled.

## 42.3 External signal description

The FMC has no external (off-chip) signals.

## 42.4 Memory map and register descriptions

The MCM's programming model provides control and configuration of the FMC's features.

For details, see the description of the MCM's Platform Control Register (PLACR).

## 42.5 Functional description

The FMC is a flash acceleration unit with flexible buffers for user configuration.

Besides managing the interface between bus masters and the program flash memory, the FMC can be used to customize the program flash memory cache and buffer to provide single-cycle system clock data access times. Whenever a hit occurs for the prefetch speculation buffer or the cache (when enabled), the requested data is transferred within a single system clock.

Upon system reset, the FMC is configured as follows:

- Flash cache is enabled.
- Instruction speculation and caching are enabled.

- Data speculation is disabled.
- Data caching is enabled.

Though the default configuration provides flash acceleration, advanced users may desire to customize the FMC buffer configurations to maximize throughput for their use cases. For example, the user may adjust the controls to enable buffering per access type (data or instruction).

#### **NOTE**

When reconfiguring the FMC, do not program the control and configuration inputs to the FMC while the program flash memory is being accessed. Instead, change them with a routine executing from RAM in supervisor mode.



# Chapter 43

## Flash Memory Module (FTFA)

### 43.1 Introduction

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.

## 43.1.1 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.

### 43.1.1.1 Program Flash Memory Features

- Sector size of 1 KB
- Program flash protection scheme prevents accidental program or erase of stored data
- Automated, built-in, program and erase algorithms with verify

### 43.1.1.2 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

## 43.1.2 Block Diagram

The block diagram of the flash memory module is shown in the following figure.

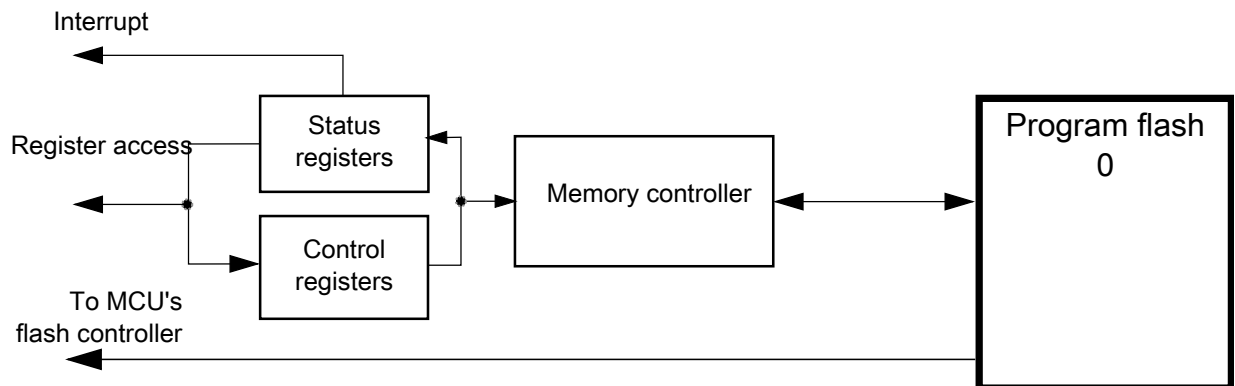


Figure 43-1. Flash Block Diagram



### 43.1.3 Glossary

**Command write sequence** — A series of MCU writes to the flash FCCOB register group that initiates and controls the execution of flash algorithms that are built into the flash memory module.

**Endurance** — The number of times that a flash memory location can be erased and reprogrammed.

**FCCOB (Flash Common Command Object)** — A group of flash registers that are used to pass command, address, data, and any associated parameters to the memory controller in the flash memory module.

**Flash block** — A macro within the flash memory module which provides the nonvolatile memory storage.

**Flash Memory Module** — All flash blocks plus a flash management unit providing high-level control and an interface to MCU buses.

**IFR** — Nonvolatile information register found in each flash block, separate from the main memory array.

**Longword** — 32 bits of data with an aligned longword having byte-address[1:0] = 00.

**NVM** — Nonvolatile memory. A memory technology that maintains stored data during power-off. The flash array is an NVM using NOR-type flash memory technology.

**NVM Normal Mode** — An NVM mode that provides basic user access to flash memory module resources. The CPU or other bus masters initiate flash program and erase operations (or other flash commands) using writes to the FCCOB register group in the flash memory module.

**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.

**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.

## 43.2 External Signal Description

The flash memory module contains no signals that connect off-chip.

## 43.3 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.

### 43.3.1 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).

*Table continues on the next page...*

Flash Configuration Field Byte Address	Size (Bytes)	Field Description
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).

### 43.3.2 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 – 0xBF	192	Reserved
0xC0 – 0xFF	64	Program Once Field

#### 43.3.2.1 Program Once Field

The Program Once Field in the program flash IFR provides 64 bytes of user data storage separate from the program flash main array. The user can program the Program Once Field one time only as there is no program flash IFR erase mechanism available to the user. The Program Once Field can be read any number of times. This section of the program flash IFR is accessed in 4-byte records using the Read Once and Program Once commands (see [Read Once Command](#) and [Program Once Command](#)).

### 43.3.3 Register Descriptions

The flash memory module contains a set of memory-mapped control and status registers.

The FCCOB register group uses a big endian addressing convention. These registers can be accessed as individual bytes, aligned 16-bit half-words or aligned 32-bit words. While accessing these registers as half-words or words in a little endian system, ensure that the correct order of the individual FCCOB registers is achieved.

**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="#">43.3.3.1/821</a>
4002_0001	Flash Configuration Register (FTFA_FCNFG)	8	R/W	00h	<a href="#">43.3.3.2/822</a>
4002_0002	Flash Security Register (FTFA_FSEC)	8	R	Undefined	<a href="#">43.3.3.3/824</a>
4002_0003	Flash Option Register (FTFA_FOPT)	8	R	Undefined	<a href="#">43.3.3.4/825</a>
4002_0004	Flash Common Command Object Registers (FTFA_FCCOB3)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_0005	Flash Common Command Object Registers (FTFA_FCCOB2)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_0006	Flash Common Command Object Registers (FTFA_FCCOB1)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_0007	Flash Common Command Object Registers (FTFA_FCCOB0)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_0008	Flash Common Command Object Registers (FTFA_FCCOB7)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_0009	Flash Common Command Object Registers (FTFA_FCCOB6)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_000A	Flash Common Command Object Registers (FTFA_FCCOB5)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_000B	Flash Common Command Object Registers (FTFA_FCCOB4)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_000C	Flash Common Command Object Registers (FTFA_FCCOB3)	8	R/W	00h	<a href="#">43.3.3.5/826</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_000D	Flash Common Command Object Registers (FTFA_FCCOBA)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_000E	Flash Common Command Object Registers (FTFA_FCCOB9)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_000F	Flash Common Command Object Registers (FTFA_FCCOB8)	8	R/W	00h	<a href="#">43.3.3.5/826</a>
4002_0010	Program Flash Protection Registers (FTFA_FPROT3)	8	R/W	Undefined	<a href="#">43.3.3.6/827</a>
4002_0011	Program Flash Protection Registers (FTFA_FPROT2)	8	R/W	Undefined	<a href="#">43.3.3.6/827</a>
4002_0012	Program Flash Protection Registers (FTFA_FPROT1)	8	R/W	Undefined	<a href="#">43.3.3.6/827</a>
4002_0013	Program Flash Protection Registers (FTFA_FPROT0)	8	R/W	Undefined	<a href="#">43.3.3.6/827</a>

### 43.3.3.1 Flash Status Register (FTFA\_FSTAT)

The FSTAT register reports the operational status of the flash memory module.

The CCIF, RDCOLERR, ACCERR, and FPVIOL bits are readable and writable. The MGSTAT0 bit is read only. The unassigned bits read 0 and are not writable.

#### NOTE

When set, the Access Error (ACCERR) and Flash Protection Violation (FPVIOL) bits in this register prevent the launch of any more commands until the flag is cleared (by writing a one to it).

Address: 4002\_0000h base + 0h offset = 4002\_0000h

Bit	7	6	5	4	3	2	1	0
Read	CCIF	RDCOLERR	ACCERR	FPVIOL		0		MGSTAT0
Write	w1c	w1c	w1c	w1c				
Reset	0	0	0	0	0	0	0	0

#### FTFA\_FSTAT field descriptions

Field	Description
7 CCIF	Command Complete Interrupt Flag  Indicates that a flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command, and CCIF stays low until command completion or command violation.

*Table continues on the next page...*

## FTFA\_FSTAT field descriptions (continued)

Field	Description
	<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 ACCERR while CCIF is set. 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 FPVIOL while CCIF is set. 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> <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>

### 43.3.3.2 Flash Configuration Register (FTFA\_FCNFG)

This register provides information on the current functional state of the flash memory module.

The erase control bits (ERSAREQ and ERSSUSP) have write restrictions. The unassigned bits read as noted and are not writable.

Address: 4002\_0000h base + 1h offset = 4002\_0001h

Bit	7	6	5	4	3	2	1	0
Read	CCIE	RDCOLLIE	ERSAREQ	ERSSUSP	0	0	0	0
Write								
Reset	0	0	0	0	0	0	0	0

### FTFA\_FCENFG 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> <p>0 No request or request complete 1 Request to: <ul style="list-style-type: none"> <li>1. run the Erase All Blocks command,</li> <li>2. verify the erased state,</li> <li>3. program the security byte in the Flash Configuration Field to the unsecure state, and</li> <li>4. release MCU security by setting the FSEC[SEC] field to the unsecure state.</li> </ul> </p>
4 ERSSUSP	<p>Erase Suspend</p> <p>Allows the user to suspend (interrupt) the Erase Flash Sector command while it is executing.</p> <p>0 No suspend requested 1 Suspend the current Erase Flash Sector command execution.</p>
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. This read-only field is reserved and always has the value 0.</p>
1 Reserved	<p>This field is reserved. This read-only field is reserved and always has the value 0.</p>

Table continues on the next page...

**FTFA\_FCNFG field descriptions (continued)**

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

**43.3.3.3 Flash Security Register (FTFA\_FSEC)**

This read-only register holds all bits associated with the security of the MCU and flash memory module.

During the reset sequence, the register is loaded with the contents of the flash security byte in the Flash Configuration Field located in program flash memory. The flash basis for the values is signified by X in the reset value.

Address: 4002\_0000h base + 2h offset = 4002\_0002h

Bit	7	6	5	4	3	2	1	0
Read	KEYEN		MEEN		FSLACC		SEC	
Write	x*		x*		x*		x*	
Reset	x*	x*	x*	x*	x*	x*	x*	x*

\* Notes:

- x = Undefined at reset.

**FTFA\_FSEC field descriptions**

Field	Description
7–6 KEYEN	Backdoor Key Security Enable Enables or disables backdoor key access to the flash memory module.  00 Backdoor key access disabled 01 Backdoor key access disabled (preferred KEYEN state to disable backdoor key access) 10 Backdoor key access enabled 11 Backdoor key access disabled
5–4 MEEN	Mass Erase Enable Enables and disables mass erase capability of the flash memory module. 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	Freescall Failure Analysis Access Code

*Table continues on the next page...*



## FTFA\_FSEC field descriptions (continued)

Field	Description
	<p>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.</p> <p>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.</p> <p>00 Freescale factory access granted  01 Freescale factory access denied  10 Freescale factory access denied  11 Freescale factory access granted</p>
SEC	<p>Flash Security</p> <p>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.</p> <p>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.</p>

## 43.3.3.4 Flash Option Register (FTFA\_FOPT)

The flash option register allows the MCU to customize its operations by examining the state of these read-only bits, which are loaded from NVM at reset. The function of the bits is defined in the device's Chip Configuration details.

All bits in the register are read-only .

During the reset sequence, the register is loaded from the flash nonvolatile option byte in the Flash Configuration Field located in program flash memory. The flash basis for the values is signified by X in the reset value. However, the register is written to 0xFF if the contents of the flash nonvolatile option byte are 0x00.

Address: 4002\_0000h base + 3h offset = 4002\_0003h

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

\* Notes:

- x = Undefined at reset.

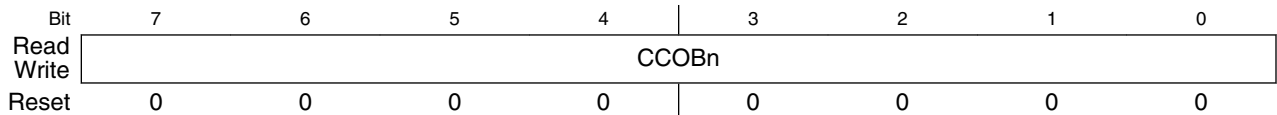
**FTFA\_FOPT field descriptions**

Field	Description
OPT	Nonvolatile Option  These bits are loaded from flash to this register at reset. Refer to the device's Chip Configuration details for the definition and use of these bits.

**43.3.3.5 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



**FTFA\_FCCOBn field descriptions**

Field	Description												
CCOBn	<p>The FCCOB register provides a command code and relevant parameters to the memory controller. The individual registers that compose the FCCOB data set can be written in any order, but you must provide all needed values, which vary from command to command. First, set up all required FCCOB fields and then initiate the command's execution by writing a 1 to the FSTAT[CCIF] bit. This clears the CCIF bit, which locks all FCCOB parameter fields and they cannot be changed by the user until the command completes (CCIF returns to 1). No command buffering or queueing is provided; the next command can be loaded only after the current command completes.</p> <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="width: 20%;">FCCOB Number</th> <th style="width: 80%;">Typical Command Parameter Contents [7:0]</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>FCMD (a code that defines the flash command)</td> </tr> <tr> <td>1</td> <td>Flash address [23:16]</td> </tr> <tr> <td>2</td> <td>Flash address [15:8]</td> </tr> <tr> <td>3</td> <td>Flash address [7:0]</td> </tr> <tr> <td>4</td> <td>Data Byte 0</td> </tr> </tbody> </table>	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
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												

## FTFA\_FCCOBN field descriptions (continued)

Field	Description	
	FCCOB Number	Typical Command Parameter Contents [7:0]
	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
	<p><b>FCCOB Endianness and Multi-Byte Access :</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. The FCCOB register group may be read and written as individual bytes, aligned words (2 bytes) or aligned longwords (4 bytes).</p>	

### 43.3.3.6 Program Flash Protection Registers (FTFA\_FPROTn)

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. 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:

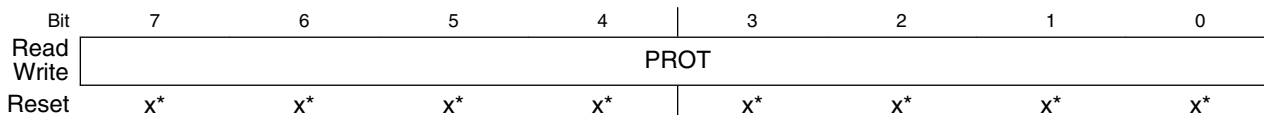
Program flash protection register	Program flash protection bits
FPROT0	PROT[31:24]
FPROT1	PROT[23:16]
FPROT2	PROT[15:8]
FPROT3	PROT[7:0]

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
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> <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 memory configurations with less than 32 KB of program flash where each assigned bit protects 1 KB .</p> <p>0 Program flash region is protected.                      1 Program flash region is not protected</p>

## 43.4 Functional Description

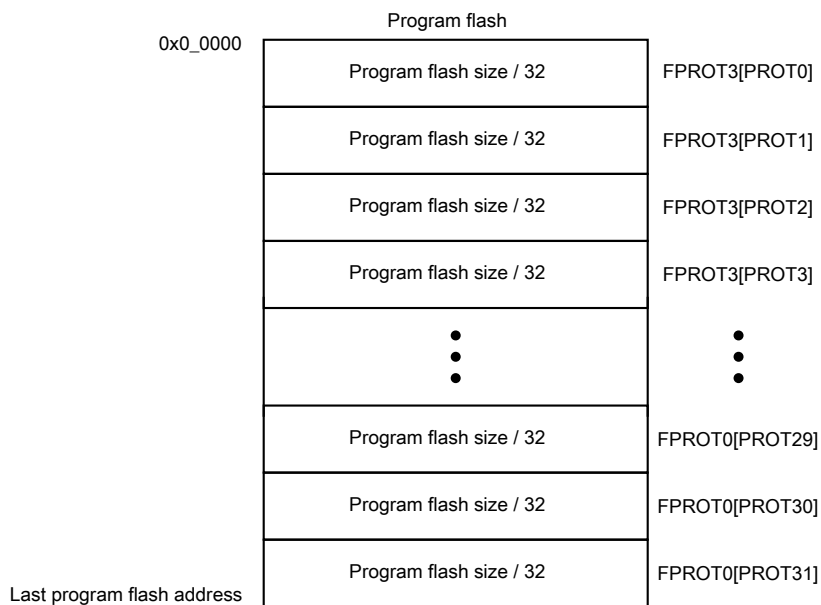
The information found here describes functional details of the flash memory module.

### 43.4.1 Flash Protection

Individual regions within the flash memory can be protected from program and erase operations.

Protection is controlled by the following registers:

- $FPROT_n$  —
  - For  $2^n$  program flash sizes, four registers typically protect 32 regions of the program flash memory as shown in the following figure



**Figure 43-2. Program flash protection**

#### NOTE

Flash protection features are discussed further in [AN4507: Using the Kinetis Security and Flash Protection Features](#). Not all features described in the application note are available on this device.

## 43.4.2 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 43-1. Flash Interrupt Sources**

Flash Event	Readable Status Bit	Interrupt Enable Bit
Flash Command Complete	FSTAT[CCIF]	FCNFG[CCIE]
Flash Read Collision Error	FSTAT[RDCOLERR]	FCNFG[RDCOLLIE]

### Note

Vector addresses and their relative interrupt priority are determined at the MCU level.

Some devices also generate a bus error response as a result of a Read Collision Error event. See the chip configuration information to determine if a bus error response is also supported.

## 43.4.3 Flash Operation in Low-Power Modes

### 43.4.3.1 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](#)).

### 43.4.3.2 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.

**43.4.4 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 43-2](#)). Refer to the Chip Configuration details of this device for how to activate each mode.

**43.4.5 Flash Reads and Ignored Writes**

The flash memory module requires only the flash address to execute a flash memory read.

The MCU must not read from the flash memory while commands are running (as evidenced by CCIF=0) on that block. Read data cannot be guaranteed from a flash block while any command is processing within that block. The block arbitration logic detects any simultaneous access and reports this as a read collision error (see the FSTAT[RDCOLERR] bit).

**43.4.6 Read While Write (RWW)**

The following simultaneous accesses are not allowed:

- Reading from program flash memory space while a flash command is active (CCIF=0).

**43.4.7 Flash Program and Erase**

All flash functions except read require the user to setup and launch a flash command through a series of peripheral bus writes.

The user cannot initiate any further flash commands until notified that the current command has completed. The flash command structure and operation are detailed in [Flash Command Operations](#).

## 43.4.8 Flash Command Operations

Flash command operations are typically used to modify flash memory contents.

The next sections describe:

- The command write sequence used to set flash command parameters and launch execution
- A description of all flash commands available

### 43.4.8.1 Command Write Sequence

Flash commands are specified using a command write sequence illustrated in [Figure 43-3](#). 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.

#### 43.4.8.1.1 Load the FCCOB Registers

The user must load the FCCOB registers with all parameters required by the desired flash command. The individual registers that make up the FCCOB data set can be written in any order.

#### 43.4.8.1.2 Launch the Command by Clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by clearing FSTAT[CCIF] by writing a '1' to it. FSTAT[CCIF] remains 0 until the flash command completes.

The FSTAT register contains a blocking mechanism that prevents a new command from launching (can't clear FSTAT[CCIF]) if the previous command resulted in an access error (FSTAT[ACCERR]=1) or a protection violation (FSTAT[FPVIOL]=1). In error scenarios, two writes to FSTAT are required to initiate the next command: the first write clears the error flags, the second write clears CCIF.



### 43.4.8.1.3 Command Execution and Error Reporting

The command processing has several steps:

1. The flash memory module reads the command code and performs a series of parameter checks and protection checks, if applicable, which are unique to each command.

If the parameter check fails, the FSTAT[ACCERR] (access error) flag is set. FSTAT[ACCERR] reports invalid instruction codes and out-of bounds addresses. Usually, access errors suggest that the command was not set-up with valid parameters in the FCCOB register group.

Program and erase commands also check the address to determine if the operation is requested to execute on protected areas. If the protection check fails, FSTAT[FPVIOL] (protection error) flag is set.

Command processing never proceeds to execution when the parameter or protection step fails. Instead, command processing is terminated after setting FSTAT[CCIF].

2. If the parameter and protection checks pass, the command proceeds to execution. Run-time errors, such as failure to erase verify, may occur during the execution phase. Run-time errors are reported in FSTAT[MGSTAT0]. A command may have access errors, protection errors, and run-time errors, but the run-time errors are not seen until all access and protection errors have been corrected.
3. Command execution results, if applicable, are reported back to the user via the FCCOB and FSTAT registers.
4. The flash memory module sets FSTAT[CCIF] signifying that the command has completed.

The flow for a generic command write sequence is illustrated in the following figure.

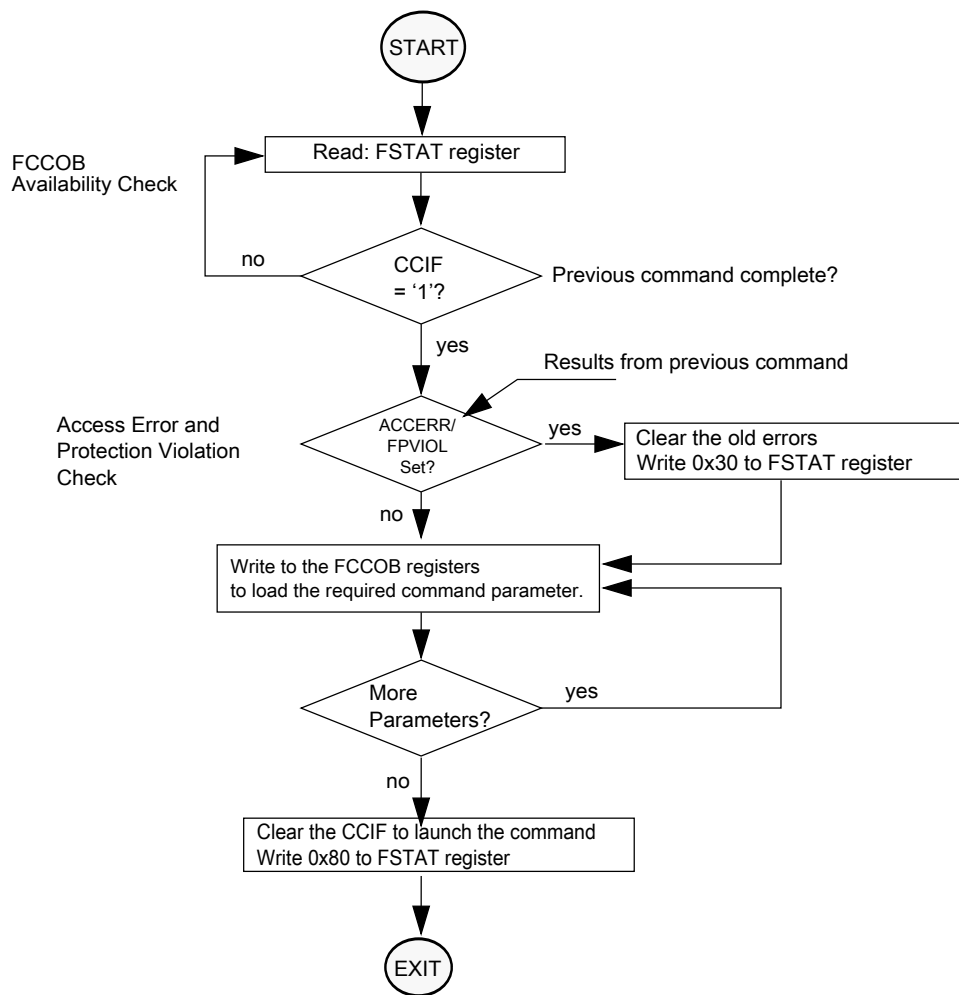


Figure 43-3. Generic flash command write sequence flowchart

### 43.4.8.2 Flash Commands

The following table summarizes the function of all flash commands.

FCMD	Command	Program flash	Function
0x01	Read 1s Section	x	Verify that a given number of program flash locations from a starting address are erased.
0x02	Program Check	x	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	x	Program 4 bytes in a program flash block.

Table continues on the next page...

FCMD	Command	Program flash	Function
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.
0x49	Erase All Blocks Unsecure	×	Erase the program flash block, verify-erase, program security byte to unsecure state, release MCU security.

### 43.4.8.3 Flash Commands by Mode

The following table shows the flash commands that can be executed in each flash operating mode.

**Table 43-2. Flash Commands by Mode**

FCMD	Command	NVM Normal			NVM Special		
		Unsecure	Secure	MEEN=10	Unsecure	Secure	MEEN=10
0x01	Read 1s Section	×	×	×	×	—	—
0x02	Program Check	×	×	×	×	—	—
0x03	Read Resource	×	×	×	×	—	—
0x06	Program Longword	×	×	×	×	—	—
0x09	Erase Flash Sector	×	×	×	×	—	—
0x40	Read 1s All Blocks	×	×	×	×	×	—
0x41	Read Once	×	×	×	×	—	—
0x43	Program Once	×	×	×	×	—	—
0x44	Erase All Blocks	×	×	×	×	×	—

*Table continues on the next page...*

**Table 43-2. Flash Commands by Mode (continued)**

FCMD	Command	NVM Normal			NVM Special		
		Unsecure	Secure	MEEN=10	Unsecure	Secure	MEEN=10
0x45	Verify Backdoor Access Key	x	x	x	x	—	—
0x49	Erase All Blocks Unsecure	x	x	x	x	x	—

### 43.4.9 Margin Read Commands

The Read-1s commands (Read 1s All Blocks, 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. Basic flash array reads use the standard, un-margined, read reference level.

Only the 'normal' read level should be employed during normal flash usage. The non-standard, 'user' and 'factory' margin levels should be employed only in special cases. They can be used during special diagnostic routines to gain confidence that the device is not suffering from the end-of-life data loss customary of flash memory devices.

Erased ('1') and programmed ('0') bit states can degrade due to elapsed time and data cycling (number of times a bit is erased and re-programmed). The lifetime of the erased states is relative to the last erase operation. The lifetime of the programmed states is measured from the last program time.

The 'user' and 'factory' levels become, in effect, a minimum safety margin; i.e. if the reads pass at the tighter tolerances of the 'user' and 'factory' margins, then the 'normal' reads have at least this much safety margin before they experience data loss.

The 'user' margin is a small delta to the normal read reference level. 'User' margin levels can be employed to check that flash memory contents have adequate margin for normal level read operations. If unexpected read results are encountered when checking flash memory contents at the 'user' margin levels, loss of information might soon occur during 'normal' readout.

The 'factory' margin is a bigger deviation from the norm, a more stringent read criteria that should only be attempted immediately (or very soon) after completion of an erase or program command, early in the cycling life. 'Factory' margin levels can be used to check that flash memory contents have adequate margin for long-term data retention at the

normal level setting. If unexpected results are encountered when checking flash memory contents at 'factory' margin levels, the flash memory contents should be erased and reprogrammed.

### CAUTION

Factory margin levels must only be used during verify of the initial factory programming.

## 43.4.10 Flash Command Description

This section describes all flash commands that can be launched by a command write sequence.

The flash memory module sets the FSTAT[ACCERR] bit and aborts the command execution if any of the following illegal conditions occur:

- There is an unrecognized command code in the FCCOB FCMD field.
- There is an error in a FCCOB field for the specific commands. Refer to the error handling table provided for each command.

Ensure that FSTAT[ACCERR] and FSTAT[FPVIOL] are cleared prior to starting the command write sequence. As described in [Launch the Command by Clearing CCIF](#), a new command cannot be launched while these error flags are set.

Do not attempt to read a flash block while the flash memory module is running a command (FSTAT[CCIF] = 0) on that same block. The flash memory module may return invalid data to the MCU with the collision error flag (FSTAT[RDCOLERR]) set.

### CAUTION

Flash data must be in the erased state before being programmed. Cumulative programming of bits (adding more zeros) is not allowed.

### 43.4.10.1 Read 1s Section Command

The Read 1s Section command checks if a section of program flash memory is erased to the specified read margin level. The Read 1s Section command defines the starting address and the number of longwords to be verified.

**Table 43-3. Read 1s Section Command FCCOB Requirements**

FCCOB Number	FCCOB Contents [7:0]
0	0x01 (RD1SEC)
1	Flash address [23:16] of the first longword to be verified
2	Flash address [15:8] of the first longword to be verified
3	Flash address [7:0] <sup>1</sup> of the first longword to be verified
4	Number of longwords to be verified [15:8]
5	Number of longwords to be verified [7:0]
6	Read-1 Margin Choice

1. Must be longword aligned (Flash address [1:0] = 00).

Upon clearing CCIF to launch the Read 1s Section command, the flash memory module sets the read margin for 1s according to [Table 43-4](#) 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 43-4. 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 43-5. 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 longword aligned.	FSTAT[ACCERR]
The requested section crosses a Flash block boundary.	FSTAT[ACCERR]
The requested number of longwords is 0.	FSTAT[ACCERR]
Read-1s fails.	FSTAT[MGSTAT0]

### 43.4.10.2 Program Check Command

The Program Check command tests a previously programmed program flash longword to see if it reads correctly at the specified margin level.

**Table 43-6. 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 43-7](#), 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 43-7. 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 43-8. 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]

*Table continues on the next page...*

**Table 43-8. Program Check Command Error Handling (continued)**

Error Condition	Error Bit
Flash address is not longword aligned	FSTAT[ACCERR]
An invalid margin choice is supplied	FSTAT[ACCERR]
Either of the margin reads does not match the expected data	FSTAT[MGSTAT0]

### 43.4.10.3 Read Resource Command

The Read Resource command allows the user to read data from special-purpose memory resources located within the flash memory module. The special-purpose memory resources available include program flash IFR space and the Version ID field. Each resource is assigned a select code as shown in [Table 43-10](#).

**Table 43-9. 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 43-10</a> )

1. Must be longword aligned (Flash address [1:0] = 00).

**Table 43-10. 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 43-11. 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]

#### 43.4.10.4 Program Longword Command

The Program Longword command programs four previously-erased bytes in the program flash memory using an embedded algorithm.

#### CAUTION

A flash memory location must be in the erased state before being programmed. Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash memory location is not allowed. Re-programming of existing 0s to 0 is not allowed as this overstresses the device.

**Table 43-12. 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 43-13. Program Longword Command Error Handling**

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
An invalid flash address is supplied	FSTAT[ACCERR]
Flash address is not longword aligned	FSTAT[ACCERR]
Flash address points to a protected area	FSTAT[FPVIOL]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

### 43.4.10.5 Erase Flash Sector Command

The Erase Flash Sector operation erases all addresses in a flash sector.

**Table 43-14. 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 longword aligned (flash address [1:0] = 00).

After clearing CCIF to launch the Erase Flash Sector command, the flash memory module erases the selected program flash 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 43-4](#)).

**Table 43-15. 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 longword aligned	FSTAT[ACCERR]
The selected program flash sector 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 Section command to verify all bits are erased.

#### 43.4.10.5.1 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.

#### 43.4.10.5.2 Resuming a Suspended Erase Flash Sector Operation

If the ERSSUSP bit is still set when CCIF is cleared to launch the next command, the previous Erase Flash Sector operation resumes. The flash memory module acknowledges the request to resume a suspended operation by clearing the ERSSUSP bit. A new suspend request can then be made by setting ERSSUSP. A single Erase Flash Sector operation can be suspended and resumed multiple times.

There is a minimum elapsed time limit 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.

### 43.4.10.5.3 Aborting a Suspended Erase Flash Sector Operation

The user may choose to abort a suspended Erase Flash Sector operation by clearing the ERSSUSP bit prior to clearing CCIF for the next command launch. When a suspended operation is aborted, the flash memory module starts the new command using the new FCCOB contents.

#### Note

Aborting the erase leaves the bitcells in an indeterminate, partially-erased state. Data in this sector is not reliable until a new erase command fully completes.

The following figure shows how to suspend and resume the Erase Flash Sector operation.

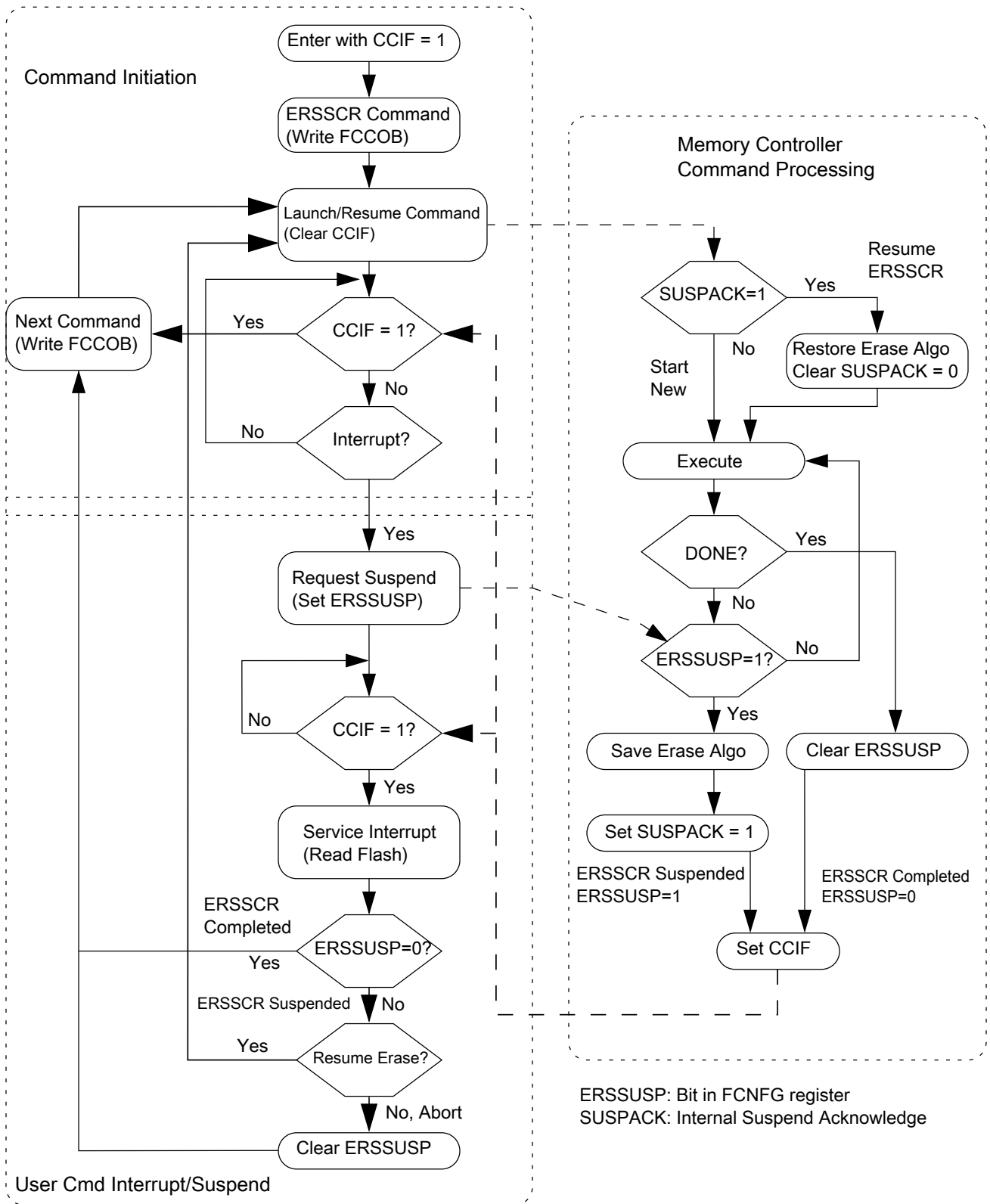


Figure 43-4. Suspend and Resume of Erase Flash Sector Operation

### 43.4.10.6 Read 1s All Blocks Command

The Read 1s All Blocks command checks if the program flash blocks have been erased to the specified read margin level, if applicable, and releases security if the readout passes, i.e. all data reads as '1'.

**Table 43-16. 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 43-17](#),
- checks the contents of the program flash are in the erased state.

If the flash memory module confirms that these memory resources are erased, security is released by setting the FSEC[SEC] field to the unsecure state. The security byte in the flash configuration field (see [Flash Configuration Field Description](#)) remains unaffected by the Read 1s All Blocks command. If the read fails, i.e. all memory resources are not in the fully erased state, the FSTAT[MGSTAT0] bit is set.

The CCIF flag sets after the Read 1s All Blocks operation has completed.

**Table 43-17. 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 43-18. Read 1s All Blocks Command Error Handling**

Error Condition	Error Bit
An invalid margin choice is specified	FSTAT[ACCERR]
Read-1s fails	FSTAT[MGSTAT0]

### 43.4.10.7 Read Once Command

The Read Once command provides read access to special 64-byte fields located in the program flash 0 IFR (see [Program Flash IFR Map](#) and [Program Once Field](#)). Access to the Program Once field is via 16 records (index values 0x00 - 0x0F), each 4 bytes long. These fields are programmed using the Program Once command described in [Program Once Command](#).

**Table 43-19. Read Once Command FCCOB Requirements**

FCCOB Number	FCCOB Contents [7:0]
0	0x41 (RDONCE)
1	Program Once record index (0x00 - 0x0F)
2	Not used
3	Not used
Returned Values	
4	Program Once byte 0 value
5	Program Once byte 1 value
6	Program Once byte 2 value
7	Program Once byte 3 value

After clearing CCIF to launch the Read Once command, a 4-byte Program Once record is read 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 - 0x0F. 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 43-20. 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]

### 43.4.10.8 Program Once Command

The Program Once command enables programming to special 64-byte fields in the program flash 0 IFR (see [Program Flash IFR Map](#) and [Program Once Field](#)). Access to the Program Once field is via 16 records (index values 0x00 - 0x0F), each 4 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 43-21. Program Once Command FCCOB Requirements**

FCCOB Number	FCCOB Contents [7:0]
0	0x43 (PGMONCE)
1	Program Once record index (0x00 - 0x0F)
2	Not Used
3	Not Used
4	Program Once byte 0 value
5	Program Once byte 1 value
6	Program Once byte 2 value
7	Program Once byte 3 value

After clearing CCIF to launch the Program Once command, the flash memory module first verifies that the selected record is erased. If erased, then the selected record is programmed using the values provided. The Program Once command also verifies that the programmed values read back correctly. The CCIF flag is set after the Program Once operation has completed.

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 - 0x0F. 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 43-22. 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]
Any errors have been encountered during the verify operation	FSTAT[MGSTAT0]

1. If a Program Once record is initially programmed to 0xFFFF\_FFFF, the Program Once command is allowed to execute again on that same record.

### 43.4.10.9 Erase All Blocks Command

The Erase All Blocks operation erases all flash memory, verifies all memory contents, and releases MCU security.

**Table 43-23. 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, 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.

**Table 43-24. 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.

#### 43.4.10.9.1 Triggering an Erase All External to the Flash Memory Module

The functionality of the Erase All Blocks/Erase 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, 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](#).

#### 43.4.10.10 Verify Backdoor Access Key Command

The Verify Backdoor Access Key command only executes if the mode and security conditions are satisfied (see [Flash Commands by Mode](#)). Execution of the Verify Backdoor Access Key command is further qualified by the FSEC[KEYEN] bits. The Verify Backdoor Access Key command releases security if user-supplied keys in the FCCOB match those stored in the Backdoor Comparison Key bytes of the Flash

Configuration Field (see [Flash Configuration Field Description](#)). The column labelled Flash Configuration Field offset address shows the location of the matching byte in the Flash Configuration Field.

**Table 43-25. 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
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 43-26. 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]

### 43.4.10.11 Erase All Blocks Unsecure Command

The Erase All Blocks Unsecure operation erases all flash memory, verifies all memory contents, programs the security byte in the Flash Configuration Field to the unsecure state, and releases MCU security.

**Table 43-27. Erase All Blocks Unsecure Command FCCOB Requirements**

FCCOB Number	FCCOB Contents [7:0]
0	0x49 (ERSALLU)

After clearing CCIF to launch the Erase All Blocks Unsecure command, the flash memory module erases all program flash memory, then verifies that all are erased.

If the flash memory module verifies that all program flash memory was properly erased, security is released by setting the FSEC[SEC] field to the unsecure state, and the security byte (see [Flash Configuration Field Description](#)) is programmed to the unsecure state by the Erase All Blocks Unsecure command. If the erase or program verify fails, the FSTAT[MGSTAT0] bit is set. The CCIF flag is set after the Erase All Blocks Unsecure operation completes.

**Table 43-28. Erase All Blocks Unsecure Command Error Handling**

Error Condition	Error Bit
Command not available in current mode/security	FSTAT[ACCERR]
Any errors have been encountered during erase or program verify operations	FSTAT[MGSTAT0]

## 43.4.11 Security

The flash memory module provides security information to the MCU based on contents of the FSEC security register.

The MCU then limits access to flash memory resources as defined in the device's Chip Configuration details. During reset, the flash memory module initializes the FSEC register using data read from the security byte of the Flash Configuration Field (see [Flash Configuration Field Description](#)).

The following fields are available in the FSEC register. The settings are described in the [Flash Security Register \(FTFA\\_FSEC\)](#) details.

Flash security features are discussed further in [AN4507: Using the Kinetis Security and Flash Protection Features](#). Note that not all features described in the application note are available on this device.

**Table 43-29. FSEC register fields**

FSEC field	Description
KEYEN	Backdoor Key Access
MEEN	Mass Erase Capability
FSLACC	Freescale Factory Access
SEC	MCU security

### 43.4.11.1 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 43-30. 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, Erase All Blocks Unsecure and Read 1s All Blocks commands.

### 43.4.11.2 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.

#### 43.4.11.2.1 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.

### 43.4.12 Reset Sequence

On each system reset the flash memory module executes a sequence which establishes initial values for the flash block configuration parameters, FPROT, FOPT, 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.

---

# Appendix A

## Release Notes for Revision 2.1

### A.1 General changes throughout

- Added support of FGPIO.







**How to Reach Us:**

**Home Page:**

[nxp.com](http://nxp.com)

**Web Support:**

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, Freescale, the Freescale logo, the Energy Efficient Solutions logo, and Kinetis are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, the ARM powered logo, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2014-2016 NXP B.V.

Document Number KL13P80M48SF3RM  
Revision 2.1, 07/2016

