

**MSP432P4xx Family**

# **Technical Reference Manual**



Literature Number: SLAU356E  
March 2015–Revised December 2016

<b>Preface.....</b>	<b>39</b>
<b>1 Cortex-M4F Processor .....</b>	<b>40</b>
1.1 Introduction.....	41
1.1.1 Block Diagram .....	43
1.2 Overview.....	44
1.2.1 Bus Interface.....	44
1.2.2 Integrated Configurable Debug .....	45
1.2.3 Cortex-M4F System Component Details .....	45
1.3 Programming Model .....	46
1.3.1 Processor Mode and Privilege Levels for Software Execution .....	46
1.3.2 Stacks.....	46
1.3.3 Register Map.....	47
1.3.4 Register Descriptions .....	47
1.3.5 Exceptions and Interrupts .....	50
1.3.6 Data Types .....	50
1.4 Memory Model .....	50
1.4.1 Memory Regions, Types, and Attributes .....	50
1.4.2 Memory System Ordering of Memory Accesses .....	51
1.4.3 Behavior of Memory Accesses .....	51
1.4.4 Software Ordering of Memory Accesses .....	51
1.4.5 Bit-Banding .....	52
1.4.6 Data Storage .....	54
1.5 Exception Model.....	55
1.5.1 Exception States .....	56
1.5.2 Exception Types.....	56
1.5.3 Exception Handlers .....	57
1.5.4 Vector Table .....	58
1.5.5 Exception Priorities.....	59
1.5.6 Interrupt Priority Grouping.....	59
1.5.7 Level and Pulse Interrupts .....	59
1.5.8 Exception Entry and Return.....	60
1.6 Fault Handling .....	62
1.6.1 Fault Types.....	63
1.6.2 Fault Escalation and Hard Faults .....	63
1.6.3 Fault Status Registers and Fault Address Registers .....	64
1.6.4 Lockup .....	64
1.7 Power Management.....	64
1.8 Instruction Set Summary .....	65
<b>2 Cortex-M4F Peripherals.....</b>	<b>70</b>
2.1 Cortex-M4F Peripherals Introduction .....	71
2.2 Functional Peripherals Description.....	71
2.2.1 System Timer (SysTick) .....	71
2.2.2 Nested Vectored Interrupt Controller (NVIC) .....	72
2.2.3 System Control Block (SCB).....	73
2.2.4 Memory Protection Unit (MPU).....	74

2.2.5	Floating-Point Unit (FPU).....	78
2.3	Debug Peripherals Description .....	81
2.3.1	FPB .....	81
2.3.2	DWT.....	81
2.3.3	ITM .....	82
2.3.4	TPIU.....	82
2.4	Functional Peripherals Registers.....	83
2.4.1	FPU Registers .....	83
2.4.2	MPU Registers.....	89
2.4.3	NVIC Registers.....	105
2.4.4	SYSTICK Registers.....	120
2.4.5	SCB Registers.....	125
2.4.6	SCnSCB Registers .....	160
2.4.7	COREDEBUG Registers.....	163
2.5	Debug Peripherals Registers.....	170
2.5.1	FPB Registers .....	170
2.5.2	DWT Registers .....	181
2.5.3	ITM Registers .....	207
<b>3</b>	<b>Reset Controller (RSTCTL) .....</b>	<b>244</b>
3.1	Introduction .....	245
3.2	Reset Classification.....	245
3.2.1	Class 0 : Power On/Off Reset (POR) Class.....	245
3.2.2	Class 1 : Reboot Reset .....	246
3.2.3	Class 2 : Hard Reset .....	246
3.2.4	Class 3 : Soft Reset .....	247
3.3	RSTCTL Registers.....	248
3.3.1	RSTCTL_RESET_REQ Register (offset = 00h) .....	249
3.3.2	RSTCTL_HARDRESET_STAT Register (offset = 04h) .....	250
3.3.3	RSTCTL_HARDRESET_CLR Register (offset = 08h) .....	251
3.3.4	RSTCTL_HARDRESET_SET Register (offset = 0Ch).....	252
3.3.5	RSTCTL_SOFTRESET_STAT Register (offset = 10h).....	253
3.3.6	RSTCTL_SOFTRESET_CLR Register (offset = 14h) .....	254
3.3.7	RSTCTL_SOFTRESET_SET Register (offset = 18h).....	255
3.3.8	RSTCTL_PSSRESET_STAT Register (offset = 100h) .....	256
3.3.9	RSTCTL_PSSRESET_CLR Register (offset = 104h).....	256
3.3.10	RSTCTL_PCMRESET_STAT Register (offset = 108h).....	257
3.3.11	RSTCTL_PCMRESET_CLR Register (offset = 10Ch).....	257
3.3.12	RSTCTL_PINRESET_STAT Register (offset = 110h) .....	258
3.3.13	RSTCTL_PINRESET_CLR Register (offset = 114h).....	258
3.3.14	RSTCTL_REBOOTRESET_STAT Register (offset = 118h) .....	259
3.3.15	RSTCTL_REBOOTRESET_CLR Register (offset = 11Ch) .....	259
3.3.16	RSTCTL_CSRESET_STAT Register (offset = 120h) .....	260
3.3.17	RSTCTL_CSRESET_CLR Register (offset = 124h) .....	260
<b>4</b>	<b>System Controller (SYSCTL) .....</b>	<b>261</b>
4.1	SYSCTL Introduction .....	262
4.2	Device Memory Configuration and Status .....	262
4.2.1	Flash .....	262
4.2.2	SRAM .....	262
4.3	NMI Configuration .....	263
4.4	Watchdog Timer Reset Configuration.....	263
4.5	Peripheral Halt Control .....	263
4.6	Glitch Filtering on Digital I/Os .....	263
4.7	Reset Status and Override Control.....	264

4.8	Device Security .....	264
4.8.1	Device Security Introduction.....	264
4.8.2	Device Security Components .....	264
4.8.3	JTAG and SWD Lock Based Security .....	264
4.8.4	IP Protection Through Secure Memory Zones.....	264
4.8.5	In-Field Updates .....	266
4.8.6	Boot-Overrides .....	268
4.8.7	Device Security and Boot Overrides User Considerations .....	278
4.9	Device Descriptor Table .....	279
4.9.1	Tag Length Value (TLV) Descriptors .....	280
4.9.2	TLV Checksum.....	280
4.9.3	Calibration Values .....	281
4.10	ARM Cortex-M4F ROM Table Based Part Number .....	284
4.11	SYSCTL Registers.....	285
4.11.1	SYS_REBOOT_CTL Register (offset = 0000h) .....	286
4.11.2	SYS_NMI_CTLSTAT Register (offset = 0004h) .....	287
4.11.3	SYS_WDTRESET_CTL Register (offset = 0008h) .....	288
4.11.4	SYS_PERIHALT_CTL Register (offset = 000Ch) .....	289
4.11.5	SYS_SRAM_SIZE Register (offset = 0010h) .....	290
4.11.6	SYS_SRAM_BANKEN Register (offset = 0014h) .....	291
4.11.7	SYS_SRAM_BANKRET Register (offset = 0018h) .....	292
4.11.8	SYS_FLASH_SIZE Register (offset = 0020h) .....	293
4.11.9	SYS_DIO_GLTFLT_CTL Register (offset = 0030h) .....	294
4.11.10	SYS_SECDATA_UNLOCK Register (offset = 0040h) .....	295
4.11.11	SYS_MASTER_UNLOCK Register (offset = 1000h) .....	296
4.11.12	SYS_BOOTOVER_REQ0 Register (offset = 1004h).....	297
4.11.13	SYS_BOOTOVER_REQ1 Register (offset = 1008h).....	298
4.11.14	SYS_BOOTOVER_ACK Register (offset = 100Ch) .....	299
4.11.15	SYS_RESET_REQ Register (offset = 1010h) .....	300
4.11.16	SYS_RESET_STATOVER Register (offset = 1014h) .....	301
4.11.17	SYS_SYSTEM_STAT Register (offset = 1020h) .....	302
<b>5</b>	<b>Clock System (CS) .....</b>	<b>303</b>
5.1	Clock System Introduction .....	304
5.2	Clock System Operation .....	306
5.2.1	CS Module Features for Low-Power Applications .....	306
5.2.2	LFXT Oscillator .....	306
5.2.3	HFXT Oscillator .....	307
5.2.4	Internal Very-Low-Power Low-Frequency Oscillator (VLO) .....	308
5.2.5	Internal Low-Power Low-Frequency Oscillator (REFO) .....	309
5.2.6	Module Oscillator (MODOSC).....	310
5.2.7	System Oscillator (SYSOSC) .....	310
5.2.8	Digitally Controlled Oscillator (DCO) .....	310
5.2.9	Module Clock Request System .....	312
5.2.10	Clock System Fail-Safe Operation .....	314
5.2.11	Start-Up Counters .....	316
5.2.12	Synchronization of Clock Signals.....	316
5.2.13	Clock Status .....	317
5.3	CS Registers .....	318
5.3.1	CSKEY Register (offset = 00h) [reset = 0000_A596h].....	319
5.3.2	CSCTL0 Register (offset = 04h) [reset = 0001_0000h].....	320
5.3.3	CSCTL1 Register (offset = 08h) [reset = 0000_0033h].....	321
5.3.4	CSCTL2 Register (offset = 0Ch) [reset = 0001_0003h] .....	323
5.3.5	CSCTL3 Register (offset = 10h) [reset = 0000_00BBh] .....	325

5.3.6	CSCLKEN Register (offset = 30h) [reset = 0000_000Fh] .....	326
5.3.7	CSSTAT Register (offset = 34h) [reset = 0000_0003h] .....	327
5.3.8	CSIE Register (offset = 40h) [reset = 0000_0000h] .....	329
5.3.9	CSIFG Register (offset = 48h) [reset = 0000_0001h] .....	330
5.3.10	CSCLRIFG Register (offset = 50h) [reset = 0000_0000h] .....	331
5.3.11	CSSETIFG Register (offset = 58h) [reset = 0000_0000h] .....	332
5.3.12	CSDCOERCAL0 Register (offset = 60h) [reset = 0100_0000h] .....	333
5.3.13	CSDCOERCAL1 Register (offset = 64h) [reset = 0000_0100h] .....	334
<b>6</b>	<b>Power Supply System (PSS) .....</b>	<b>335</b>
6.1	Power Supply System (PSS) Introduction .....	336
6.2	PSS Operation .....	337
6.2.1	Supply Voltage Supervisor / Monitor .....	337
6.2.2	Supply Voltage Supervisor during Power-Up .....	338
6.2.3	VCCDET .....	338
6.2.4	PSS Interrupts.....	339
6.3	PSS Registers.....	340
6.3.1	PSSKEY Register (offset = 00h) [reset = 0000A596h] .....	341
6.3.2	PSSCTL0 Register (offset = 04h) [reset = 00002000h].....	342
6.3.3	PSSIE Register (offset = 34h) [reset = 0000h].....	344
6.3.4	PSSIFG Register (offset = 38h) [reset = 0000h].....	345
6.3.5	PSSCLRIFG Register (offset = 3Ch) [reset = 0000h].....	346
<b>7</b>	<b>Power Control Manager (PCM) .....</b>	<b>347</b>
7.1	PCM Introduction .....	348
7.2	PCM Overview .....	348
7.3	Core Voltage Regulators.....	349
7.3.1	DC-DC Regulator Care Abouts .....	349
7.4	Power Modes .....	349
7.4.1	Active Modes (AM) .....	350
7.4.2	LPM0 .....	350
7.4.3	LPM3 and LPM4 .....	351
7.4.4	LPM3.5 and LPM4.5.....	351
7.4.5	Summary of Power Modes .....	351
7.5	Power Mode Transitions .....	353
7.5.1	Active Mode Transitions .....	354
7.5.2	Transitions To and From LPM0 .....	355
7.5.3	Transitions To and From LPM3 and LPM4.....	355
7.5.4	Transitions To and From LPM3.5 and LPM4.5 .....	356
7.6	Changing Core Voltages.....	356
7.6.1	Increasing V <sub>CORE</sub> for Higher MCLK Frequencies .....	356
7.6.2	Decreasing V <sub>CORE</sub> for Power Optimization .....	356
7.7	ARM Cortex Processor Sleep Modes .....	357
7.7.1	WFI, Wait For Interrupt.....	357
7.7.2	WFE, Wait For Event.....	357
7.7.3	Sleep On Exit.....	357
7.7.4	SLEEPDEEP .....	358
7.8	Power Mode Requests .....	358
7.9	Power Mode Selection .....	358
7.10	Power Mode Transition Checks.....	359
7.11	Power Mode Clock Checks.....	359
7.12	Clock Configuration Changes .....	360
7.13	Changing Active Modes.....	360
7.13.1	DC-DC Error Checking.....	361
7.14	Entering LPM0 Modes .....	362

7.15	Exiting LPM0 Modes .....	362
7.16	Entering LPM3 or LPM4 Modes .....	362
7.17	Exiting LPM3 or LPM4 Modes .....	363
7.18	Entering LPM3.5 and LPM4.5 Modes .....	363
7.19	Exiting LPM3.5 and LPM4.5 Modes .....	364
7.20	Supply Voltage Supervisor and Monitor and Power Modes.....	365
7.21	Low-Power Reset .....	366
7.22	Power Requests During Debug .....	366
7.22.1	Debug During Active Modes .....	366
7.22.2	Debug During LPM0 Modes.....	366
7.22.3	Debug During LPM3, LPM4, and LPMx.5 Modes .....	367
7.23	Wake-up Sources From Low Power Modes.....	367
7.24	PCM Registers .....	368
7.24.1	PCMCTL0 Register (offset = 00h) [reset = A5960000h].....	369
7.24.2	PCMCTL1 Register (offset = 04h) [reset = A5960000h].....	371
7.24.3	PCMIE Register (offset = 08h) [reset = 00000000h] .....	373
7.24.4	PCMIFG Register (offset = 0Ch) [reset = 00000000h] .....	374
7.24.5	PCMCLRIFG Register (offset = 10h) [reset = 00000000h].	375
<b>8</b>	<b>Flash Controller (FLCTL) .....</b>	<b>376</b>
8.1	Introduction .....	377
8.1.1	Flash Memory Organization .....	377
8.1.2	Flash Controller Address Mapping .....	377
8.1.3	Flash Controller Access Privileges .....	377
8.2	Common Operations Using the Flash Controller .....	378
8.2.1	Using MSP432 Driver Library for Flash Operations.....	378
8.2.2	Flash Read .....	378
8.2.3	Flash Program.....	379
8.2.4	Flash Erase .....	379
8.2.5	Flash Program and Erase Protection .....	380
8.3	Advanced Operations using the Flash Controller.....	380
8.3.1	Advanced Flash Read .....	380
8.3.2	Advanced Flash Program.....	381
8.3.3	Advanced Flash Erase .....	391
8.3.4	Flash Controller Interrupts .....	392
8.3.5	Application Benchmarking Features.....	393
8.3.6	Support for AM_LF_VCOREx and LPM0_LF_VCOREx Power Modes .....	393
8.3.7	Flash Functionality During Resets .....	393
8.4	FLCTL Registers .....	395
8.4.1	FLCTL_POWER_STAT Register (offset = 0000h) .....	397
8.4.2	FLCTL_BANK0_RDCTL Register (offset = 0010h) .....	398
8.4.3	FLCTL_BANK1_RDCTL Register (offset = 0014h) .....	400
8.4.4	FLCTL_RDBRST_CTLSTAT Register (offset = 0020h) .....	402
8.4.5	FLCTL_RDBRST_STARTADDR Register (offset = 0024h) .....	403
8.4.6	FLCTL_RDBRST_LEN Register (offset = 0028h) .....	404
8.4.7	FLCTL_RDBRST_FAILADDR Register (offset = 003Ch) .....	405
8.4.8	FLCTL_RDBRST_FAILCNT Register (offset = 0040h) .....	406
8.4.9	FLCTL_PRG_CTLSTAT Register (offset = 0050h) .....	407
8.4.10	FLCTL_PRGBRST_CTLSTAT Register (offset = 0054h) .....	408
8.4.11	FLCTL_PRGBRST_STARTADDR Register (offset = 0058h) .....	410
8.4.12	FLCTL_PRGBRST_DATA0_0 Register (offset = 060h) .....	411
8.4.13	FLCTL_PRGBRST_DATA0_1 Register (offset = 064h) .....	411
8.4.14	FLCTL_PRGBRST_DATA0_2 Register (offset = 068h) .....	412
8.4.15	FLCTL_PRGBRST_DATA0_3 Register (offset = 06Ch) .....	412

8.4.16	FLCTL_PRGBRST_DATA1_0 Register (offset = 070h) .....	413
8.4.17	FLCTL_PRGBRST_DATA1_1 Register (offset = 074h) .....	413
8.4.18	FLCTL_PRGBRST_DATA1_2 Register (offset = 078h) .....	414
8.4.19	FLCTL_PRGBRST_DATA1_3 Register (offset = 07Ch) .....	414
8.4.20	FLCTL_PRGBRST_DATA2_0 Register (offset = 080h) .....	415
8.4.21	FLCTL_PRGBRST_DATA2_1 Register (offset = 084h) .....	415
8.4.22	FLCTL_PRGBRST_DATA2_2 Register (offset = 088h) .....	416
8.4.23	FLCTL_PRGBRST_DATA2_3 Register (offset = 08Ch) .....	416
8.4.24	FLCTL_PRGBRST_DATA3_0 Register (offset = 090h) .....	417
8.4.25	FLCTL_PRGBRST_DATA3_1 Register (offset = 094h) .....	417
8.4.26	FLCTL_PRGBRST_DATA3_2 Register (offset = 098h) .....	418
8.4.27	FLCTL_PRGBRST_DATA3_3 Register (offset = 09Ch) .....	418
8.4.28	FLCTL_ERASE_CTLSTAT Register (offset = 00A0h) .....	419
8.4.29	FLCTL_ERASE_SECTADDR Register (offset = 00A4h) .....	420
8.4.30	FLCTL_BANK0_INFO_WEPROT Register (offset = 00B0h) .....	421
8.4.31	FLCTL_BANK0_MAIN_WEPROT Register (offset = 00B4h) .....	422
8.4.32	FLCTL_BANK1_INFO_WEPROT Register (offset = 00C0h) .....	424
8.4.33	FLCTL_BANK1_MAIN_WEPROT Register (offset = 00C4h) .....	425
8.4.34	FLCTL_BMRK_CTLSTAT Register (offset = 00D0h) .....	427
8.4.35	FLCTL_BMRK_IFETCH Register (offset = 00D4h) .....	428
8.4.36	FLCTL_BMRK_DREAD Register (offset = 00D8h) .....	429
8.4.37	FLCTL_BMRK_CMP Register (offset = 00DCh) .....	430
8.4.38	FLCTL_IFG Register (offset = 0F0h) .....	431
8.4.39	FLCTL_IE Register (offset = 0F4h) .....	432
8.4.40	FLCTL_CLRIFG Register (offset = 0F8h) .....	433
8.4.41	FLCTL_SETIFG Register (offset = 0FCCh) .....	434
8.4.42	FLCTL_READ_TIMCTL Register (offset = 0100h) .....	435
8.4.43	FLCTL_READMARGIN_TIMCTL Register (offset = 0104h) .....	436
8.4.44	FLCTL_PRGVER_TIMCTL Register (offset = 0108h) .....	437
8.4.45	FLCTL_ERSVER_TIMCTL Register (offset = 010Ch) .....	438
8.4.46	FLCTL_PROGRAM_TIMCTL Register (offset = 0114h) .....	439
8.4.47	FLCTL_ERASE_TIMCTL Register (offset = 0118h) .....	440
8.4.48	FLCTL_MASSEREASE_TIMCTL Register (offset = 011Ch) .....	441
8.4.49	FLCTL_BURSTPRG_TIMCTL Register (offset = 0120h) .....	442
<b>9</b>	<b>DMA .....</b>	<b>443</b>
9.1	DMA Introduction .....	444
9.2	DMA Operation.....	445
9.2.1	APB Slave Interface .....	445
9.2.2	AHB Master Interface .....	445
9.2.3	DMA Control Interface .....	447
9.2.4	Channel Control Data Structure .....	461
9.2.5	Peripheral Triggers .....	468
9.2.6	Interrupts .....	468
9.3	DMA Registers .....	469
9.3.1	DMA_DEVICE_CFG Register (offset = 000h) .....	470
9.3.2	DMA_SW_CHTRIG Register (offset = 004h) .....	471
9.3.3	DMA_CHn_SRCCFG Register (offset = 010h + 4h*n, n = 0 through NUM_DMA_CHANNELS) .....	473
9.3.4	DMA_INT1_SRCCFG Register (offset = 100h) .....	474
9.3.5	DMA_INT2_SRCCFG Register (offset = 104h) .....	475
9.3.6	DMA_INT3_SRCCFG Register (offset = 108h) .....	476
9.3.7	DMA_INT0_SRCFLG Register (offset = 110h) .....	477
9.3.8	DMA_INT0_CLRFLG Register (offset = 114h) .....	478
9.3.9	DMA_STAT Register (offset = 1000h) [reset = 0h].....	480

9.3.10	DMA_CFG Register (offset = 1004h) [reset = 0h] .....	481
9.3.11	DMA_CTLBASE Register (offset = 1008h) [reset = 0h] .....	482
9.3.12	DMA_ALTBASE Register (offset = 100Ch) [reset = 0h] .....	483
9.3.13	DMA_WAITSTAT Register (offset = 1010h) [reset = 0h] .....	484
9.3.14	DMA_SWREQ Register (offset = 1014h) [reset = 0h] .....	485
9.3.15	DMA_USEBURSTSET Register (offset = 1018h) [reset = 0h] .....	486
9.3.16	DMA_USEBURSTCLR Register (offset = 101Ch) [reset = 0h] .....	487
9.3.17	DMA_REQMASKSET Register (offset = 1020h) [reset = 0h] .....	488
9.3.18	DMA_REQMASKCLR Register (offset = 1024h) [reset = 0h] .....	489
9.3.19	DMA_ENASET Register (offset = 1028h) [reset = 0h] .....	490
9.3.20	DMA_ENACLR Register (offset = 102Ch) [reset = 0h] .....	491
9.3.21	DMA_ALTSET Register (offset = 1030h) [reset = 0h] .....	492
9.3.22	DMA_ALTCLR Register (offset = 1034h) [reset = 0h] .....	493
9.3.23	DMA_PRIOSET Register (offset = 1038h) [reset = 0h] .....	494
9.3.24	DMA_PRIOCLR Register (offset = 103Ch) [reset = 0h] .....	495
9.3.25	DMA_ERRCLR Register (offset = 104Ch) [reset = 0h] .....	496
<b>10</b>	<b>Digital I/O .....</b>	<b>497</b>
10.1	Digital I/O Introduction .....	498
10.2	Digital I/O Operation .....	499
10.2.1	Input Registers (PxIN) .....	499
10.2.2	Output Registers (PxOUT) .....	499
10.2.3	Direction Registers (PxDIR) .....	499
10.2.4	Pullup or Pulldown Resistor Enable Registers (PxREN) .....	499
10.2.5	Output Drive Strength Selection Registers (PxDS) .....	500
10.2.6	Function Select Registers (PxSEL0, PxSEL1) .....	500
10.2.7	Port Interrupts .....	500
10.3	I/O Configuration .....	502
10.3.1	Configuration After Reset .....	502
10.3.2	Configuration of Unused Port Pins .....	502
10.3.3	Configuration for LPM3 and LPM4 Modes .....	502
10.3.4	Configuration for LPM3.5 and LPM4.5 Modes .....	502
10.4	Digital I/O Registers .....	504
10.4.1	PxIV Register .....	516
10.4.2	PxIN Register .....	517
10.4.3	PxOUT Register .....	517
10.4.4	PxDIR Register .....	517
10.4.5	PxREN Register .....	518
10.4.6	PxDS Register .....	518
10.4.7	PxSEL0 Register .....	518
10.4.8	PxSEL1 Register .....	519
10.4.9	PxSELC Register .....	519
10.4.10	PxIES Register .....	519
10.4.11	PxIE Register .....	520
10.4.12	PxIFG Register .....	520
<b>11</b>	<b>Port Mapping Controller (PMAP) .....</b>	<b>521</b>
11.1	Port Mapping Controller Introduction .....	522
11.2	Port Mapping Controller Operation .....	522
11.2.1	Access .....	522
11.2.2	Mapping .....	522
11.3	PMAP Registers .....	525
11.3.1	PMAPKEYID Register (offset = 00h) [reset = 96A5h] .....	527
11.3.2	PMAPCTL Register (offset = 02h) [reset = 0001h] .....	527
11.3.3	P1MAP0 to P1MAP7 Register (offset = 08h to 0Fh) [reset = X] .....	527

11.3.4	P2MAP0 to P2MAP7 Register (offset = 10h to 17h) [reset = X].....	528
11.3.5	P3MAP0 to P3MAP7 Register (offset = 18h to 1Fh) [reset = X] .....	528
11.3.6	P4MAP0 to P4MAP7 Register (offset = 20h to 27h) [reset = X].....	528
11.3.7	P5MAP0 to P5MAP7 Register (offset = 28h to 2Fh) [reset = X] .....	528
11.3.8	P6MAP0 to P6MAP7 Register (offset = 30h to 37h) [reset = X].....	529
11.3.9	P7MAP0 to P7MAP7 Register (offset = 38h to 3Fh) [reset = X] .....	529
11.3.10	PxMAPyz Register [reset = X].....	529
<b>12</b>	<b>Capacitive Touch IO (CAPTIO).....</b>	<b>530</b>
12.1	Capacitive Touch IO Introduction.....	531
12.2	Capacitive Touch IO Operation .....	532
12.3	CapTouch Registers.....	533
12.3.1	CAPTIOxCTL Register (offset = 0Eh) [reset = 0000h].....	534
<b>13</b>	<b>CRC32 Module.....</b>	<b>535</b>
13.1	Cyclic Redundancy Check (CRC32) Module Introduction.....	536
13.2	CRC Checksum Generation .....	536
13.2.1	CRC Standard and Bit Order.....	537
13.2.2	CRC Implementation .....	537
13.3	CRC32 Registers .....	538
<b>14</b>	<b>AES256 Accelerator .....</b>	<b>549</b>
14.1	AES Accelerator Introduction.....	550
14.2	AES Accelerator Operation.....	551
14.2.1	Load the Key (128-Bit, 192-Bit, or 256-Bit Keylength).....	552
14.2.2	Load the Data (128-Bit State) .....	552
14.2.3	Read the Data (128-Bit State) .....	553
14.2.4	Trigger an Encryption or Decryption .....	553
14.2.5	Encryption .....	554
14.2.6	Decryption .....	554
14.2.7	Decryption Key Generation.....	555
14.2.8	AES Key Buffer .....	556
14.2.9	Using the AES Accelerator With Low-Power Modes.....	557
14.2.10	AES Accelerator Interrupts.....	557
14.2.11	DMA Operation and Implementing Block Cipher Modes.....	557
14.3	AES256 Registers .....	567
14.3.1	AESACTL0 Register.....	568
14.3.2	AESACTL1 Register.....	570
14.3.3	AESASTAT Register .....	571
14.3.4	AESAKEY Register.....	572
14.3.5	AESADIN Register .....	573
14.3.6	AESADOUT Register .....	574
14.3.7	AESAXDIN Register.....	575
14.3.8	AESAXIN Register.....	576
<b>15</b>	<b>Watchdog Timer (WDT_A).....</b>	<b>577</b>
15.1	WDT_A Introduction .....	578
15.2	WDT_A Operation .....	580
15.2.1	Watchdog Timer Counter (WDTcnt).....	580
15.2.2	Watchdog Mode .....	580
15.2.3	Interval Timer Mode .....	580
15.2.4	Watchdog Related Interrupts and Flags .....	581
15.2.5	Clock Sources of the WDT_A.....	581
15.2.6	WDT_A Operation in Different Device Power Modes.....	581
15.3	WDT_A Registers.....	583
15.3.1	WDTCTL Register .....	584

<b>16</b>	<b>Timer32 .....</b>	<b>585</b>
16.1	Introduction .....	586
16.2	Functional Description.....	586
16.3	Operation .....	586
16.4	Interrupt Generation .....	587
16.5	Timer32 Registers .....	588
16.5.1	T32LOAD1 Register (offset = 00h) [reset = 0h] .....	589
16.5.2	T32VALUE1 Register (offset = 04h) [reset = FFFFFFFFh] .....	590
16.5.3	T32CONTROL1 Register (offset = 08h) [reset = 20h] .....	591
16.5.4	T32INTCLR1 Register (offset = 0Ch) [reset = undefined] .....	592
16.5.5	T32RIS1 Register (offset = 10h) [reset = 0h] .....	593
16.5.6	T32MIS1 Register (offset = 14h) [reset = 0h] .....	594
16.5.7	T32BGLOAD1 Register (offset = 18h) [reset = 0h].....	595
16.5.8	T32LOAD2 Register (offset = 20h) [reset = 0h] .....	596
16.5.9	T32VALUE2 Register (offset = 24h) [reset = FFFFFFFFh] .....	597
16.5.10	T32CONTROL2 Register (offset = 28h) [reset = 20h].....	598
16.5.11	T32INTCLR2 Register (offset = 2Ch) [reset = undefined] .....	599
16.5.12	T32RIS2 Register (offset = 30h) [reset = 0h] .....	600
16.5.13	T32MIS2 Register (offset = 34h) [reset = 0h] .....	601
16.5.14	T32BGLOAD2 Register (offset = 38h) [reset = 0h] .....	602
<b>17</b>	<b>Timer_A .....</b>	<b>603</b>
17.1	Timer_A Introduction .....	604
17.2	Timer_A Operation .....	606
17.2.1	16-Bit Timer Counter .....	606
17.2.2	Starting the Timer.....	606
17.2.3	Timer Mode Control .....	607
17.2.4	Capture/Compare Blocks .....	610
17.2.5	Output Unit .....	612
17.2.6	Timer_A Interrupts.....	616
17.3	Timer_A Registers .....	617
17.3.1	TAxCTL Register .....	618
17.3.2	TAxR Register.....	619
17.3.3	TAxCCTL0 to TAxCCTL6 Register.....	620
17.3.4	TAxCCR0 to TAxCCR6 Register .....	622
17.3.5	TAxIV Register .....	622
17.3.6	TAxEX0 Register .....	623
<b>18</b>	<b>Real-Time Clock (RTC_C) .....</b>	<b>624</b>
18.1	RTC_C Introduction .....	625
18.2	RTC_C Operation.....	627
18.2.1	Calendar Mode.....	627
18.2.2	Real-Time Clock and Prescale Dividers .....	627
18.2.3	Real-Time Clock Alarm Function .....	627
18.2.4	Real-Time Clock Protection .....	628
18.2.5	Reading or Writing Real-Time Clock Registers .....	628
18.2.6	Real-Time Clock Interrupts .....	629
18.2.7	Real-Time Clock Calibration for Crystal Offset Error.....	630
18.2.8	Real-Time Clock Compensation for Crystal Temperature Drift.....	630
18.2.9	Real-Time Clock Operation in Low-Power Modes .....	633
18.3	RTC_C Registers .....	634
18.3.1	RTCCCTL0_L Register .....	636
18.3.2	RTCCCTL0_H Register.....	637
18.3.3	RTCCCTL1 Register .....	638
18.3.4	RTCCCTL3 Register .....	639

---

18.3.5	RTCOCAL Register .....	640
18.3.6	RTCTCMP Register .....	641
18.3.7	RTCSEC Register – Hexadecimal Format .....	642
18.3.8	RTCSEC Register – BCD Format .....	642
18.3.9	RTCMIN Register – Hexadecimal Format .....	643
18.3.10	RTCMIN Register – BCD Format .....	643
18.3.11	RTCHOUR Register – Hexadecimal Format .....	644
18.3.12	RTCHOUR Register – BCD Format .....	644
18.3.13	RTCDOW Register .....	645
18.3.14	RTCDAY Register – Hexadecimal Format.....	645
18.3.15	RTCDAY Register – BCD Format.....	645
18.3.16	RTCMON Register – Hexadecimal Format .....	646
18.3.17	RTCMON Register – BCD Format .....	646
18.3.18	RTCYEAR Register – Hexadecimal Format .....	647
18.3.19	RTCYEAR Register – BCD Format.....	647
18.3.20	RTCAMIN Register – Hexadecimal Format .....	648
18.3.21	RTCAMIN Register – BCD Format .....	648
18.3.22	RTCAHOUR Register – Hexadecimal Format .....	649
18.3.23	RTCAHOUR Register – BCD Format .....	649
18.3.24	RTCADOW Register – Calendar Mode .....	650
18.3.25	RTCADAY Register – Hexadecimal Format .....	650
18.3.26	RTCADAY Register – BCD Format.....	650
18.3.27	RTCPSS0CTL Register .....	651
18.3.28	RTCPSS1CTL Register .....	652
18.3.29	RTCPSS0 Register .....	653
18.3.30	RTCPSS1 Register .....	653
18.3.31	RTCIV Register.....	654
18.3.32	RTCBIN2BCD Register.....	655
18.3.33	RTCBIN2BIN Register.....	655
<b>19</b>	<b>Reference Module (REF_A) .....</b>	<b>656</b>
19.1	REF_A Introduction.....	657
19.2	Principle of Operation .....	658
19.2.1	Low-Power Operation .....	658
19.2.2	Reference System Requests.....	658
19.3	REF_A Registers .....	660
19.3.1	REFCTL0 Register (offset = 00h) [reset = 0008h] .....	661
<b>20</b>	<b>ADC14 .....</b>	<b>663</b>
20.1	ADC14 Introduction.....	664
20.2	ADC14 Operation .....	665
20.2.1	14-Bit ADC Core .....	666
20.2.2	ADC14 Inputs and Multiplexer .....	666
20.2.3	Voltage References .....	667
20.2.4	Auto Power Down .....	667
20.2.5	Power Modes.....	667
20.2.6	Sample and Conversion Timing .....	667
20.2.7	Conversion Memory .....	670
20.2.8	ADC14 Conversion Modes .....	671
20.2.9	Window Comparator.....	676
20.2.10	Using the Integrated Temperature Sensor.....	677
20.2.11	ADC14 Grounding and Noise Considerations.....	678
20.2.12	ADC14 Calibration .....	679
20.2.13	ADC14 Interrupts.....	679
20.3	ADC14 Registers .....	680

20.3.1	ADC14CTL0 Register (offset = 00h) [reset = 00000000h] .....	681
20.3.2	ADC14CTL1 Register (offset = 04h) [reset = 00000030h] .....	684
20.3.3	ADC14LO0 Register (offset = 08h) [reset = 00000000h] .....	686
20.3.4	ADC14HI0 Register (offset = 0Ch) [reset = 00003FFFh] .....	687
20.3.5	ADC14LO1 Register (offset = 10h) [reset = 00000000h] .....	688
20.3.6	ADC14HI1 Register (offset = 14h) [reset = 00003FFFh] .....	689
20.3.7	ADC14MCTL0 to ADC14MCTL31 Register (offset = 018h to 094h) [reset = 00000000h] .....	690
20.3.8	ADC14MEM0 to ADC14MEM31 Register (offset = 098h to 104h) [reset = Undefined] .....	692
20.3.9	ADC14IER0 Register (offset = 13Ch) [reset = 00000000h] .....	693
20.3.10	ADC14IER1 Register (offset = 140h) [reset = 00000000h] .....	696
20.3.11	ADC14IFGR0 Register (offset = 144h) [reset = 00000000h] .....	697
20.3.12	ADC14IFGR1 Register (offset = 148h) [reset = 00000000h] .....	701
20.3.13	ADC14CLRIFGR0 Register (offset = 14Ch) [reset = 00000000h] .....	702
20.3.14	ADC14CLRIFGR1 Register (offset = 150h) [reset = 00000000h] .....	705
20.3.15	ADC14IV Register (offset = 154h) [reset = 00000000h] .....	706
<b>21</b>	<b>Comparator E Module (COMP_E) .....</b>	<b>708</b>
21.1	COMP_E Introduction .....	709
21.2	COMP_E Operation .....	710
21.2.1	Comparator .....	710
21.2.2	Analog Input Switches .....	710
21.2.3	Port Logic .....	710
21.2.4	Input Short Switch .....	710
21.2.5	Output Filter .....	711
21.2.6	Reference Voltage Generator .....	712
21.2.7	Port Disable Register (CEPD) .....	713
21.2.8	Comparator_E Interrupts .....	713
21.2.9	Comparator_E Used to Measure Resistive Elements .....	713
21.3	COMP_E Registers .....	716
21.3.1	CExCTL0 Register (offset = 00h) [reset = 0000h] .....	717
21.3.2	CExCTL1 Register (offset = 02h) [reset = 0000h] .....	718
21.3.3	CExCTL2 Register (offset = 04h) [reset = 0000h] .....	719
21.3.4	CExCTL3 Register (offset = 06h) [reset = 0000h] .....	720
21.3.5	CExINT Register (offset = 0Ch) [reset = 0000h] .....	722
21.3.6	CExIV Register (offset = 0Eh) [reset = 0000h] .....	723
<b>22</b>	<b>Enhanced Universal Serial Communication Interface (eUSCI) – UART Mode .....</b>	<b>724</b>
22.1	Enhanced Universal Serial Communication Interface A (eUSCI_A) Overview .....	725
22.2	eUSCI_A Introduction – UART Mode .....	725
22.3	eUSCI_A Operation – UART Mode .....	727
22.3.1	eUSCI_A Initialization and Reset .....	727
22.3.2	Character Format .....	727
22.3.3	Asynchronous Communication Format .....	727
22.3.4	Automatic Baud-Rate Detection .....	730
22.3.5	IrDA Encoding and Decoding .....	731
22.3.6	Automatic Error Detection .....	732
22.3.7	eUSCI_A Receive Enable .....	733
22.3.8	eUSCI_A Transmit Enable .....	733
22.3.9	UART Baud-Rate Generation .....	734
22.3.10	Setting a Baud Rate .....	736
22.3.11	Transmit Bit Timing - Error calculation .....	737
22.3.12	Receive Bit Timing – Error Calculation .....	737
22.3.13	Typical Baud Rates and Errors .....	738
22.3.14	Using the eUSCI_A Module in UART Mode With Low-Power Modes .....	740
22.3.15	eUSCI_A Interrupts .....	740

22.3.16	DMA Operation .....	742
22.4	eUSCI_A UART Registers.....	743
22.4.1	UCAxCTLW0 Register .....	744
22.4.2	UCAxCTLW1 Register .....	745
22.4.3	UCAxBRW Register .....	746
22.4.4	UCAxMCTLW Register .....	746
22.4.5	UCAxSTATW Register .....	747
22.4.6	UCAxRXBUF Register .....	748
22.4.7	UCAxTXBUF Register .....	748
22.4.8	UCAxABCTL Register.....	749
22.4.9	UCAxIRCTL Register.....	750
22.4.10	UCAxIE Register .....	751
22.4.11	UCAxIFG Register .....	752
22.4.12	UCAxIV Register .....	753
<b>23</b>	<b>Enhanced Universal Serial Communication Interface (eUSCI) – SPI Mode .....</b>	<b>754</b>
23.1	Enhanced Universal Serial Communication Interfaces (eUSCI_A, eUSCI_B) Overview .....	755
23.2	eUSCI Introduction – SPI Mode .....	755
23.3	eUSCI Operation – SPI Mode.....	757
23.3.1	eUSCI Initialization and Reset .....	757
23.3.2	Character Format .....	758
23.3.3	Master Mode .....	758
23.3.4	Slave Mode .....	759
23.3.5	SPI Enable.....	760
23.3.6	Serial Clock Control .....	760
23.3.7	Using the SPI Mode With Low-Power Modes.....	761
23.3.8	SPI Interrupts.....	761
23.4	eUSCI_A SPI Registers.....	763
23.4.1	UCAxCTLW0 Register .....	764
23.4.2	UCAxBRW Register .....	765
23.4.3	UCAxSTATW Register .....	766
23.4.4	UCAxRXBUF Register .....	767
23.4.5	UCAxTXBUF Register .....	768
23.4.6	UCAxIE Register.....	769
23.4.7	UCAxIFG Register.....	770
23.4.8	UCAxIV Register.....	771
23.5	eUSCI_B SPI Registers.....	772
23.5.1	UCBxCTLW0 Register .....	773
23.5.2	UCBxBRW Register .....	774
23.5.3	UCBxSTATW Register .....	774
23.5.4	UCBxRXBUF Register .....	775
23.5.5	UCBxTXBUF Register .....	775
23.5.6	UCBxE Register .....	776
23.5.7	UCBxIFG Register.....	776
23.5.8	UCBxIV Register.....	777
<b>24</b>	<b>Enhanced Universal Serial Communication Interface (eUSCI) – I<sup>2</sup>C Mode .....</b>	<b>778</b>
24.1	Enhanced Universal Serial Communication Interface B (eUSCI_B) Overview .....	779
24.2	eUSCI_B Introduction – I <sup>2</sup> C Mode .....	779
24.3	eUSCI_B Operation – I <sup>2</sup> C Mode .....	780
24.3.1	eUSCI_B Initialization and Reset .....	781
24.3.2	I <sup>2</sup> C Serial Data .....	781
24.3.3	I <sup>2</sup> C Addressing Modes .....	782
24.3.4	I <sup>2</sup> C Module Operating Modes .....	783
24.3.5	Glitch Filtering .....	793

---

24.3.6	I <sup>2</sup> C Clock Generation and Synchronization.....	793
24.3.7	Byte Counter .....	795
24.3.8	Multiple Slave Addresses.....	795
24.3.9	Using the eUSCI_B Module in I <sup>2</sup> C Mode With Low-Power Modes .....	796
24.3.10	eUSCI_B Interrupts in I <sup>2</sup> C Mode .....	796
24.4	eUSCI_B I <sup>2</sup> C Registers .....	799
24.4.1	UCBxCTLW0 Register .....	800
24.4.2	UCBxCTLW1 Register .....	802
24.4.3	UCBxBRW Register .....	804
24.4.4	UCBxSTATW.....	804
24.4.5	UCBxTBCNT Register .....	805
24.4.6	UCBxRXBUF Register .....	806
24.4.7	UCBxTXBUF .....	806
24.4.8	UCBxI2COA0 Register.....	807
24.4.9	UCBxI2COA1 Register.....	808
24.4.10	UCBxI2COA2 Register .....	808
24.4.11	UCBxI2COA3 Register .....	809
24.4.12	UCBxADDRX Register .....	809
24.4.13	UCBxADDMASK Register .....	810
24.4.14	UCBxI2CSA Register .....	810
24.4.15	UCBxIE Register .....	811
24.4.16	UCBxIFG Register .....	813
24.4.17	UCBxIV Register .....	815
	<b>Revision History .....</b>	<b>816</b>

---

## List of Figures

1-1.	CPU Block Diagram.....	44
1-2.	Cortex-M4F Register Set.....	47
1-3.	Bit-Band Mapping .....	54
1-4.	Data Storage .....	55
1-5.	Vector Table .....	58
1-6.	Exception Stack Frame.....	61
2-1.	SRD Use Example .....	78
2-2.	FPU Register Bank.....	79
2-3.	TPIU Block Diagram .....	82
2-4.	FPCCR Register.....	84
2-5.	FPCAR Register.....	85
2-6.	FPDSCR Register .....	86
2-7.	MVFR0 Register.....	87
2-8.	MVFR1 Register.....	88
2-9.	TYPE Register .....	90
2-10.	CTRL Register .....	91
2-11.	RNR Register .....	92
2-12.	RBAR Register .....	93
2-13.	RASR Register .....	94
2-14.	RBAR_A1 Register.....	96
2-15.	RASR_A1 Register.....	97
2-16.	RBAR_A2 Register.....	99
2-17.	RASR_A2 Register .....	100
2-18.	RBAR_A3 Register .....	102
2-19.	RASR_A3 Register .....	103
2-20.	ISER0 Register.....	106
2-21.	ISER1 Register.....	106
2-22.	ICER0 Register .....	107
2-23.	ICER1 Register .....	107
2-24.	ISPR0 Register.....	108
2-25.	ISPR1 Register.....	108
2-26.	ICPR0 Register .....	109
2-27.	ICPR1 Register .....	109
2-28.	IABR0 Register.....	110
2-29.	IABR1 Register.....	110
2-30.	IPR0 Register .....	111
2-31.	IPR1 Register .....	111
2-32.	IPR2 Register .....	112
2-33.	IPR3 Register .....	112
2-34.	IPR4 Register .....	113
2-35.	IPR5 Register .....	113
2-36.	IPR6 Register .....	114
2-37.	IPR7 Register .....	114
2-38.	IPR8 Register .....	115
2-39.	IPR9 Register .....	115
2-40.	IPR10 Register .....	116
2-41.	IPR11 Register .....	116

2-42.	IPR12 Register .....	117
2-43.	IPR13 Register .....	117
2-44.	IPR14 Register .....	118
2-45.	IPR15 Register .....	118
2-46.	STIR Register .....	119
2-47.	STCSR Register .....	121
2-48.	STRVR Register .....	122
2-49.	STCVR Register .....	123
2-50.	STCR Register .....	124
2-51.	CPUID Register .....	126
2-52.	ICSR Register .....	127
2-53.	VTOR Register .....	129
2-54.	AIRCR Register .....	130
2-55.	SCR Register .....	132
2-56.	CCR Register .....	133
2-57.	SHPR1 Register .....	134
2-58.	SHPR2 Register .....	135
2-59.	SHPR3 Register .....	136
2-60.	SHCSR Register .....	137
2-61.	CFSR Register .....	139
2-62.	HFSR Register .....	141
2-63.	DFSR Register .....	142
2-64.	MMFAR Register .....	143
2-65.	BFAR Register .....	144
2-66.	AFSR Register .....	145
2-67.	PFR0 Register .....	146
2-68.	PFR1 Register .....	147
2-69.	DFR0 Register .....	148
2-70.	AFR0 Register .....	149
2-71.	MMFR0 Register .....	150
2-72.	MMFR1 Register .....	151
2-73.	MMFR2 Register .....	152
2-74.	MMFR3 Register .....	153
2-75.	ISAR0 Register .....	154
2-76.	ISAR1 Register .....	155
2-77.	ISAR2 Register .....	156
2-78.	ISAR3 Register .....	157
2-79.	ISAR4 Register .....	158
2-80.	CPACR Register .....	159
2-81.	ICTR Register .....	161
2-82.	ACTLR Register .....	162
2-83.	DHCSR Register .....	164
2-84.	DCRSR Register .....	166
2-85.	DCRDR Register .....	167
2-86.	DEMCR Register .....	168
2-87.	FP_CTRL Register .....	171
2-88.	FP_REMAP Register .....	172
2-89.	FP_COMP0 Register .....	173
2-90.	FP_COMP1 Register .....	174

---

2-91.	FP_COMP2 Register .....	175
2-92.	FP_COMP3 Register .....	176
2-93.	FP_COMP4 Register .....	177
2-94.	FP_COMP5 Register .....	178
2-95.	FP_COMP6 Register .....	179
2-96.	FP_COMP7 Register .....	180
2-97.	CTRL Register .....	182
2-98.	CYCCNT Register .....	184
2-99.	CPICNT Register .....	185
2-100.	EXCCNT Register .....	186
2-101.	SLEEPCNT Register .....	187
2-102.	LSUCNT Register .....	188
2-103.	FOLDCNT Register .....	189
2-104.	PCSR Register .....	190
2-105.	COMP0 Register .....	191
2-106.	MASK0 Register .....	192
2-107.	FUNCTION0 Register .....	193
2-108.	COMP1 Register .....	195
2-109.	MASK1 Register .....	196
2-110.	FUNCTION1 Register .....	197
2-111.	COMP2 Register .....	199
2-112.	MASK2 Register .....	200
2-113.	FUNCTION2 Register .....	201
2-114.	COMP3 Register .....	203
2-115.	MASK3 Register .....	204
2-116.	FUNCTION3 Register .....	205
2-117.	STIM0 Register .....	208
2-118.	STIM1 Register .....	209
2-119.	STIM2 Register .....	210
2-120.	STIM3 Register .....	211
2-121.	STIM4 Register .....	212
2-122.	STIM5 Register .....	213
2-123.	STIM6 Register .....	214
2-124.	STIM7 Register .....	215
2-125.	STIM8 Register .....	216
2-126.	STIM9 Register .....	217
2-127.	STIM10 Register .....	218
2-128.	STIM11 Register .....	219
2-129.	STIM12 Register .....	220
2-130.	STIM13 Register .....	221
2-131.	STIM14 Register .....	222
2-132.	STIM15 Register .....	223
2-133.	STIM16 Register .....	224
2-134.	STIM17 Register .....	225
2-135.	STIM18 Register .....	226
2-136.	STIM19 Register .....	227
2-137.	STIM20 Register .....	228
2-138.	STIM21 Register .....	229
2-139.	STIM22 Register .....	230

2-140. STIM23 Register .....	231
2-141. STIM24 Register .....	232
2-142. STIM25 Register .....	233
2-143. STIM26 Register .....	234
2-144. STIM27 Register .....	235
2-145. STIM28 Register .....	236
2-146. STIM29 Register .....	237
2-147. STIM30 Register .....	238
2-148. STIM31 Register .....	239
2-149. TER Register .....	240
2-150. TPR Register .....	240
2-151. TCR Register .....	241
2-152. IWR Register .....	242
2-153. IMCR Register .....	242
2-154. LAR Register .....	243
2-155. LSR Register .....	243
3-1. Reset Classes .....	245
3-2. RSTCTL_RESET_REQ Register .....	249
3-3. RSTCTL_HARDRESET_STAT Register .....	250
3-4. RSTCTL_HARDRESET_CLR Register .....	251
3-5. RSTCTL_HARDRESET_SET Register .....	252
3-6. RSTCTL_SOFTRESET_STAT Register .....	253
3-7. RSTCTL_SOFTRESET_CLR Register .....	254
3-8. RSTCTL_SOFTRESET_SET Register .....	255
3-9. RSTCTL_PSSRESET_STAT Register .....	256
3-10. RSTCTL_PSSRESET_CLR Register .....	256
3-11. RSTCTL_PCMRESET_STAT Register .....	257
3-12. RSTCTL_PCMRESET_CLR Register .....	257
3-13. RSTCTL_PINRESET_STAT Register .....	258
3-14. RSTCTL_PINRESET_CLR Register .....	258
3-15. RSTCTL_REBOOTRESET_STAT Register .....	259
3-16. RSTCTL_REBOOTRESET_CLR Register .....	259
3-17. RSTCTL_CSRESET_STAT Register .....	260
3-18. RSTCTL_CSRESET_CLR Register .....	260
4-1. IP Protected Secure Zones Representation .....	265
4-2. Data Setup for Encrypted Update .....	267
4-3. Data Setup for IP Protected Secure Zone Unencrypted Update .....	268
4-4. Boot Override Flow .....	269
4-5. Factory Reset Boot Override Command Through JTAG .....	270
4-6. Device Descriptor Table .....	279
4-7. ARM Cortex-M4 Peripheral ID Register Description .....	284
4-8. Example of ROM PID Entries for MSP432P401xx MCU .....	284
4-9. SYS_REBOOT_CTL Register .....	286
4-10. SYS_NMI_CTLSTAT Register .....	287
4-11. SYS_WDTRESET_CTL Register .....	288
4-12. SYS_PERIHALT_CTL Register .....	289
4-13. SYS_SRAM_SIZE Register .....	290
4-14. SYS_SRAM_BANKEN Register .....	291
4-15. SYS_SRAM_BANKRET Register .....	292

4-16.	SYS_FLASH_SIZE Register .....	293
4-17.	SYS_DIO_GLFLT_CTL Register.....	294
4-18.	SYS_SECDATA_UNLOCK .....	295
4-19.	SYS_MASTER_UNLOCK Register.....	296
4-20.	SYS_BOOTOVER_REQ0 Register .....	297
4-21.	SYS_BOOTOVER_REQ1 Register .....	298
4-22.	SYS_BOOTOVER_ACK Register .....	299
4-23.	SYS_RESET_REQ Register .....	300
4-24.	SYS_RESET_STATOVER Register .....	301
4-25.	SYS_SYSTEM_STAT Register .....	302
5-1.	Clock System Block Diagram .....	305
5-2.	Module Clock Request System .....	313
5-3.	Oscillator Fault Logic .....	315
5-4.	Switch MCLK from DCOCLK to LFXTCLK .....	317
5-5.	CSKEY Register .....	319
5-6.	CSCTL0 Register .....	320
5-7.	CSCTL1 Register .....	321
5-8.	CSCTL2 Register .....	323
5-9.	CSCTL3 Register .....	325
5-10.	CSCLKEN Register.....	326
5-11.	CSSTAT Register.....	327
5-12.	CSIE Register .....	329
5-13.	CSIFG Register .....	330
5-14.	CSCLRIFG Register.....	331
5-15.	CSSETIFG Register .....	332
5-16.	CSDCOERCAL0 Register .....	333
5-17.	CSDCOERCAL1 Register .....	334
6-1.	PSS Block Diagram .....	336
6-2.	Supply Voltage Failure and Resulting PSS Action .....	338
6-3.	PSS Action at Device Power-Up.....	338
6-4.	PSSKEY Register.....	341
6-5.	PSSCTL0 Register .....	342
6-6.	PSSIE Register .....	344
6-7.	PSSIFG Register.....	345
6-8.	PSSCLRIFG Register .....	346
7-1.	Power Control Manager Interaction .....	348
7-2.	High Level Power Mode Transitions .....	354
7-3.	Valid Active Mode Transitions.....	354
7-4.	Valid LPM0 Transitions .....	355
7-5.	Valid LPM3 and LPM4 Transitions .....	355
7-6.	Valid LPM3.5 and LPM4.5 Transitions.....	356
7-7.	Active mode transition flow. ....	361
7-8.	PCMCTL0 Register.....	369
7-9.	PCMCTL1 Register.....	371
7-10.	PCMIE Register .....	373
7-11.	PCMIFG Register .....	374
7-12.	PCMCLRIFG Register.....	375
8-1.	Immediate and Full Word Program Flow .....	384
8-2.	Pre-Verify Error Handling for Immediate and Full Word Program Flow .....	385

8-3.	Post-Verify Error Handling for Immediate and Full Word Program Flow .....	386
8-4.	Burst Program Flow .....	388
8-5.	Handling Auto-Verify Error Before the Burst Operation.....	389
8-6.	Handling Auto-Verify Error After the Burst Operation.....	390
8-7.	FLCTL_POWER_STAT Register .....	397
8-8.	FLCTL_BANK0_RDCTL Register .....	398
8-9.	FLCTL_BANK1_RDCTL Register .....	400
8-10.	FLCTL_RDBRST_CTLSTAT Register .....	402
8-11.	FLCTL_RDBRST_STARTADDR Register.....	403
8-12.	FLCTL_RDBRST_LEN Register.....	404
8-13.	FLCTL_RDBRST_FAILADDR Register.....	405
8-14.	FLCTL_RDBRST_FAILCNT Register.....	406
8-15.	FLCTL_PRG_CTLSTAT Register .....	407
8-16.	FLCTL_PRGBRST_CTLSTAT Register .....	408
8-17.	FLCTL_PRGBRST_STARTADDR Register.....	410
8-18.	FLCTL_PRGBRST_DATA0_0 Register .....	411
8-19.	FLCTL_PRGBRST_DATA0_1 Register .....	411
8-20.	FLCTL_PRGBRST_DATA0_2 Register .....	412
8-21.	FLCTL_PRGBRST_DATA0_3 Register .....	412
8-22.	FLCTL_PRGBRST_DATA1_0 Register .....	413
8-23.	FLCTL_PRGBRST_DATA1_1 Register .....	413
8-24.	FLCTL_PRGBRST_DATA1_2 Register .....	414
8-25.	FLCTL_PRGBRST_DATA1_3 Register .....	414
8-26.	FLCTL_PRGBRST_DATA2_0 Register .....	415
8-27.	FLCTL_PRGBRST_DATA2_1 Register .....	415
8-28.	FLCTL_PRGBRST_DATA2_2 Register .....	416
8-29.	FLCTL_PRGBRST_DATA2_3 Register .....	416
8-30.	FLCTL_PRGBRST_DATA3_0 Register .....	417
8-31.	FLCTL_PRGBRST_DATA3_1 Register .....	417
8-32.	FLCTL_PRGBRST_DATA3_2 Register .....	418
8-33.	FLCTL_PRGBRST_DATA3_3 Register .....	418
8-34.	FLCTL_ERASE_CTLSTAT Register .....	419
8-35.	FLCTL_ERASE_SECTADDR Register .....	420
8-36.	FLCTL_BANK0_INFO_WEPROT Register.....	421
8-37.	FLCTL_BANK0_MAIN_WEPROT Register .....	422
8-38.	FLCTL_BANK1_INFO_WEPROT Register.....	424
8-39.	FLCTL_BANK1_MAIN_WEPROT Register .....	425
8-40.	FLCTL_BMRK_CTLSTAT Register .....	427
8-41.	FLCTL_BMRK_IFETCH Register.....	428
8-42.	FLCTL_BMRK_DREAD Register .....	429
8-43.	FLCTL_BMRK_CMP Register .....	430
8-44.	FLCTL_IFG Register .....	431
8-45.	FLCTL_IE Register .....	432
8-46.	FLCTL_CLRIFG Register.....	433
8-47.	FLCTL_SETIFG Register.....	434
8-48.	FLCTL_READ_TIMCTL Register .....	435
8-49.	FLCTL_READMARGIN_TIMCTL Register .....	436
8-50.	FLCTL_PRGVER_TIMCTL Register .....	437
8-51.	FLCTL_ERSVER_TIMCTL Register .....	438

8-52.	FLCTL_PROGRAM_TIMCTL Register .....	439
8-53.	FLCTL_ERASE_TIMCTL Register .....	440
8-54.	FLCTL_MASSERASE_TIMCTL Register.....	441
8-55.	FLCTL_BURSTPRG_TIMCTL Register .....	442
9-1.	DMA Block Diagram .....	445
9-2.	DMA Signaling When Peripherals Use Pulse Requests.....	447
9-3.	DMA Signaling When Peripherals Use Level Requests .....	448
9-4.	Polling Flowchart.....	450
9-5.	Ping-Pong Example .....	453
9-6.	Memory Scatter-Gather Example.....	456
9-7.	Peripheral Scatter-Gather Example .....	459
9-8.	Memory Map for 32 Channels, Including the Alternate Data Structure.....	461
9-9.	Memory Map for Three DMA Channels, Including the Alternate Data Structure .....	463
9-10.	channel_cfg Bit Assignments.....	464
9-11.	DMA_DEVICE_CFG Register.....	470
9-12.	DMA_SW_CHTRIG Register.....	471
9-13.	DMA_CHn_SRCCFG Register.....	473
9-14.	DMA_INT1_SRCCFG Register .....	474
9-15.	DMA_INT2_SRCCFG Register .....	475
9-16.	DMA_INT3_SRCCFG Register .....	476
9-17.	DMA_INT0_SRCFLG Register.....	477
9-18.	DMA_INT0_CLRFLG Register .....	478
9-19.	DMA_STAT Register .....	480
9-20.	DMA_CFG Register .....	481
9-21.	DMA_CTLBASE Register.....	482
9-22.	DMA_ALTBASE Register.....	483
9-23.	DMA_WAITSTAT Register .....	484
9-24.	DMA_SWREQ Register.....	485
9-25.	DMA_USEBURSTSET Register .....	486
9-26.	DMA_USEBURSTCLR Register .....	487
9-27.	DMA_REQMASKSET Register .....	488
9-28.	DMA_REQMASKCLR Register .....	489
9-29.	DMA_ENASET Register .....	490
9-30.	DMA_ENACLR Register .....	491
9-31.	DMA_ALTSET Register .....	492
9-32.	DMA_ALTCLR Register.....	493
9-33.	DMA_PRIOSET Register .....	494
9-34.	DMA_PRIOCLR Register .....	495
9-35.	DMA_ERRCLR Register .....	496
10-1.	PxIV Register.....	516
10-2.	PxIN Register.....	517
10-3.	PxOUT Register.....	517
10-4.	PxDIR Register.....	517
10-5.	PxREN Register.....	518
10-6.	PxDIS Register.....	518
10-7.	PxSEL0 Register.....	518
10-8.	PxSEL1 Register.....	519
10-9.	PxSELC Register .....	519
10-10.	PxIES Register .....	519

10-11. PxIE Register.....	520
10-12. PxIFG Register.....	520
11-1. PMAPKEYID Register.....	527
11-2. PMAPCTL Register.....	527
11-3. P1MAP0 to P1MAP7 Register .....	527
11-4. P2MAP0 to P2MAP7 Register .....	528
11-5. P3MAP0 to P3MAP7 Register .....	528
11-6. P4MAP0 to P4MAP7 Register .....	528
11-7. P5MAP0 to P5MAP7 Register .....	528
11-8. P6MAP0 to P6MAP7 Register .....	529
11-9. P7MAP0 to P7MAP7 Register .....	529
11-10. PxMAPyz Register.....	529
12-1. Capacitive Touch IO Principle.....	531
12-2. Capacitive Touch IO Block Diagram .....	532
12-3. CAPTIOxCTL Register .....	534
13-1. LFSR Implementation of CRC-CCITT as Defined in Standard (Bit0 is MSB) .....	536
13-2. LFSR Implementation of CRC32-ISO3309 as Defined in Standard (Bit0 is MSB).....	536
13-3. CRC32DI Register .....	539
13-4. CRC32DIRB Register .....	540
13-5. CRC32INIRES_LO Register .....	541
13-6. CRC32INIRES_HI Register .....	542
13-7. CRC32RESR_LO Register .....	543
13-8. CRC32RESR_HI Register .....	544
13-9. CRC16DI Register .....	545
13-10. CRC16DIRB Register .....	546
13-11. CRC16INIRES Register.....	547
13-12. CRC16RESR Register .....	548
14-1. AES Accelerator Block Diagram .....	550
14-2. AES State Array Input and Output .....	551
14-3. AES Encryption Process for 128-Bit Key .....	554
14-4. AES Decryption Process Using AESOPx = 01 for 128-Bit Key .....	555
14-5. AES Decryption Process Using AESOPx = 10 and 11 for 128-Bit Key .....	556
14-6. ECB Encryption .....	559
14-7. ECB Decryption .....	560
14-8. CBC Encryption .....	561
14-9. CBC Decryption .....	562
14-10. OFB Encryption .....	563
14-11. OFB Decryption .....	564
14-12. CFB Encryption .....	565
14-13. CFB Decryption .....	566
14-14. AESACTL0 Register.....	568
14-15. AESACTL1 Register.....	570
14-16. AESASTAT Register .....	571
14-17. AESAKEY Register.....	572
14-18. AESADIN Register.....	573
14-19. AESADOUT Register.....	574
14-20. AESAXDIN Register.....	575
14-21. AESAXIN Register.....	576
15-1. Watchdog Timer Block Diagram .....	579

15-2.	WDTCTL Register .....	584
16-1.	Prescale Clock Enable Generation .....	587
16-2.	T32LOAD1 Register .....	589
16-3.	T32VALUE1 Register .....	590
16-4.	T32CONTROL1 Register .....	591
16-5.	T32INTCLR1 Register.....	592
16-6.	T32RIS1 Register.....	593
16-7.	T32MIS1 Register .....	594
16-8.	T32BGLOAD1 Register .....	595
16-9.	T32LOAD2 Register .....	596
16-10.	T32VALUE2 Register .....	597
16-11.	T32CONTROL2 Register .....	598
16-12.	T32INTCLR2 Register.....	599
16-13.	T32RIS2 Register.....	600
16-14.	T32MIS2 Register .....	601
16-15.	T32BGLOAD2 Register .....	602
17-1.	Timer_A Block Diagram.....	605
17-2.	Up Mode .....	607
17-3.	Up Mode Flag Setting .....	607
17-4.	Continuous Mode .....	608
17-5.	Continuous Mode Flag Setting .....	608
17-6.	Continuous Mode Time Intervals .....	608
17-7.	Up/Down Mode.....	609
17-8.	Up/Down Mode Flag Setting .....	609
17-9.	Output Unit in Up/Down Mode .....	610
17-10.	Capture Signal (SCS = 1).....	611
17-11.	Capture Cycle .....	611
17-12.	Output Example – Timer in Up Mode .....	613
17-13.	Output Example – Timer in Continuous Mode .....	614
17-14.	Output Example – Timer in Up/Down Mode .....	615
17-15.	TAxCTL Register.....	618
17-16.	TAxR Register.....	619
17-17.	TAxCCTL0 to TAxCCTL6 Register .....	620
17-18.	TAxCCR0 to TAxCCR6 Register .....	622
17-19.	TAxIV Register .....	622
17-20.	TAxEX0 Register.....	623
18-1.	RTC_C Block Diagram .....	626
18-2.	RTC_C Offset Error Calibration and Temperature Compensation .....	632
18-3.	RTCCTL0_L Register .....	636
18-4.	RTCCTL0_H Register .....	637
18-5.	RTCCTL1 Register .....	638
18-6.	RTCCTL3 Register .....	639
18-7.	RTCOCAL Register.....	640
18-8.	RTCTCMP Register .....	641
18-9.	RTCSEC Register .....	642
18-10.	RTCSEC Register .....	642
18-11.	RTCMIN Register .....	643
18-12.	RTCMIN Register .....	643
18-13.	RTCHOUR Register .....	644

18-14. RTCHOUR Register .....	644
18-15. RTCDOW Register .....	645
18-16. RTCDAY Register .....	645
18-17. RTCDAY Register .....	645
18-18. RTCMON Register.....	646
18-19. RTCMON Register.....	646
18-20. RTCYEAR Register.....	647
18-21. RTCYEAR Register.....	647
18-22. RTCAMIN Register .....	648
18-23. RTCAMIN Register .....	648
18-24. RTCAHOUR Register .....	649
18-25. RTCAHOUR Register .....	649
18-26. RTCADOW Register .....	650
18-27. RTCADAY Register.....	650
18-28. RTCADAY Register.....	650
18-29. RTCPS0CTL Register.....	651
18-30. RTCPS1CTL Register.....	652
18-31. RTCPS0 Register.....	653
18-32. RTCPS1 Register.....	653
18-33. RTCIV Register .....	654
18-34. RTCBIN2BCD Register .....	655
18-35. RTCBCD2BIN Register .....	655
19-1. REF_A Block Diagram .....	657
19-2. REFCCTL0 Register .....	661
20-1. ADC14 Block Diagram .....	665
20-2. Analog Multiplexer .....	667
20-3. Extended Sample Mode in 14-Bit Mode .....	668
20-4. Pulse Sample Mode in 14-Bit Mode.....	669
20-5. Analog Input Equivalent Circuit .....	669
20-6. Single-Channel Single-Conversion Mode .....	672
20-7. Sequence-of-Channels Mode .....	673
20-8. Repeat-Single-Channel Mode.....	674
20-9. Repeat-Sequence-of-Channels Mode.....	675
20-10. Typical Temperature Sensor Transfer Function .....	677
20-11. ADC14 Grounding and Noise Considerations .....	678
20-12. ADC14CTL0 Register .....	681
20-13. ADC14CTL1 Register .....	684
20-14. ADC14LO0 Register.....	686
20-15. ADC14HI0 Register.....	687
20-16. ADC14LO1 Register.....	688
20-17. ADC14HI1 Register.....	689
20-18. ADC14MCTL0 to ADC14MCTL31 Register .....	690
20-19. ADC14MEM0 to ADC14MEM31 Register .....	692
20-20. ADC14IER0 Register.....	693
20-21. ADC14IER1 Register.....	696
20-22. ADC14IFGR0 Register.....	697
20-23. ADC14IFGR1 Register .....	701
20-24. ADC14CLRIFGR0 Register .....	702
20-25. ADC14CLRIFGR1 Register .....	705

20-26.	ADC14IV Register .....	706
21-1.	COMP_E Block Diagram .....	709
21-2.	COMP_E Sample-And-Hold .....	711
21-3.	RC-Filter Response at the Output of the Comparator .....	712
21-4.	Reference Generator Block Diagram .....	712
21-5.	Transfer Characteristic and Power Dissipation in a CMOS Inverter/Buffer .....	713
21-6.	Temperature Measurement System.....	714
21-7.	Timing for Temperature Measurement Systems.....	714
21-8.	CExCTL0 Register.....	717
21-9.	CExCTL1 Register.....	718
21-10.	CExCTL2 Register.....	719
21-11.	CExCTL3 Register.....	720
21-12.	CExINT Register .....	722
21-13.	CExIV Register .....	723
22-1.	eUSCI_Ax Block Diagram – UART Mode (UCSYNC = 0).....	726
22-2.	Character Format .....	727
22-3.	Idle-Line Format.....	728
22-4.	Address-Bit Multiprocessor Format.....	729
22-5.	Auto Baud-Rate Detection – Break/Synch Sequence .....	730
22-6.	Auto Baud-Rate Detection – Synch Field.....	730
22-7.	UART vs IrDA Data Format .....	731
22-8.	Glitch Suppression, eUSCI_A Receive Not Started.....	733
22-9.	Glitch Suppression, eUSCI_A Activated .....	733
22-10.	BITCLK Baud-Rate Timing With UCOS16 = 0 .....	734
22-11.	Receive Error.....	738
22-12.	UCAxCTLW0 Register .....	744
22-13.	UCAxCTLW1 Register .....	745
22-14.	UCAxBRW Register .....	746
22-15.	UCAxMCTLW Register .....	746
22-16.	UCAxSTATW Register .....	747
22-17.	UCAxRXBUF Register .....	748
22-18.	UCAxTXBUF Register.....	748
22-19.	UCAxABCTL Register .....	749
22-20.	UCAxIRCTL Register.....	750
22-21.	UCAxIE Register .....	751
22-22.	UCAxIFG Register .....	752
22-23.	UCAxIV Register .....	753
23-1.	eUSCI Block Diagram – SPI Mode .....	756
23-2.	eUSCI Master and External Slave (UCSTEM = 0) .....	758
23-3.	eUSCI Slave and External Master.....	759
23-4.	eUSCI SPI Timing With UCMSB = 1 .....	761
23-5.	UCAxCTLW0 Register .....	764
23-6.	UCAxBRW Register .....	765
23-7.	UCAxSTATW Register .....	766
23-8.	UCAxRXBUF Register .....	767
23-9.	UCAxTXBUF Register.....	768
23-10.	UCAxIE Register .....	769
23-11.	UCAxIFG Register .....	770
23-12.	UCAxIV Register .....	771

23-13. UCBxCTLW0 Register .....	773
23-14. UCBxBRW Register .....	774
23-15. UCBxSTATW Register .....	774
23-16. UCBxRXBUF Register .....	775
23-17. UCBxTXBUF Register .....	775
23-18. UCBxIE Register .....	776
23-19. UCBxIFG Register .....	776
23-20. UCBxIV Register .....	777
24-1. eUSCI_B Block Diagram – I <sup>2</sup> C Mode .....	780
24-2. I <sup>2</sup> C Bus Connection Diagram .....	781
24-3. I <sup>2</sup> C Module Data Transfer .....	782
24-4. Bit Transfer on I <sup>2</sup> C Bus .....	782
24-5. I <sup>2</sup> C Module 7-Bit Addressing Format .....	782
24-6. I <sup>2</sup> C Module 10-Bit Addressing Format .....	783
24-7. I <sup>2</sup> C Module Addressing Format With Repeated START Condition .....	783
24-8. I <sup>2</sup> C Time-Line Legend .....	783
24-9. I <sup>2</sup> C Slave Transmitter Mode .....	785
24-10. I <sup>2</sup> C Slave Receiver Mode .....	786
24-11. I <sup>2</sup> C Slave 10-Bit Addressing Mode .....	787
24-12. I <sup>2</sup> C Master Transmitter Mode .....	789
24-13. I <sup>2</sup> C Master Receiver Mode .....	791
24-14. I <sup>2</sup> C Master 10-Bit Addressing Mode .....	792
24-15. Arbitration Procedure Between Two Master Transmitters .....	792
24-16. Synchronization of Two I <sup>2</sup> C Clock Generators During Arbitration .....	793
24-17. UCBxCTLW0 Register .....	800
24-18. UCBxCTLW1 Register .....	802
24-19. UCBxBRW Register .....	804
24-20. UCBxSTATW Register .....	804
24-21. UCBxTBCNT Register .....	805
24-22. UCBxRXBUF Register .....	806
24-23. UCBxTXBUF Register .....	806
24-24. UCBxI2COA0 Register .....	807
24-25. UCBxI2COA1 Register .....	808
24-26. UCBxI2COA2 Register .....	808
24-27. UCBxI2COA3 Register .....	809
24-28. UCBxADDRX Register .....	809
24-29. UCBxADDMASK Register .....	810
24-30. UCBxI2CSA Register .....	810
24-31. UCBxIE Register .....	811
24-32. UCBxIFG Register .....	813
24-33. UCBxIV Register .....	815

---

## List of Tables

1-1.	Cortex-M4F Optional Parameters Configuration in MSP432P4xx .....	42
1-2.	Cortex-M4F Bus Interfaces in MSP432P4xx .....	44
1-3.	Summary of Processor Mode, Privilege Level, and Stack Use .....	46
1-4.	PSR Register Combinations .....	49
1-5.	Memory Access Behavior .....	51
1-6.	SRAM Memory Bit-Banding Regions .....	53
1-7.	Peripheral Memory Bit-Banding Regions .....	53
1-8.	Exception Types .....	57
1-9.	Exception Return Behavior .....	62
1-10.	Faults .....	63
1-11.	Fault Status and Fault Address Registers .....	64
1-12.	Cortex-M4F Instruction Summary .....	65
2-1.	Core Peripheral Register Regions .....	71
2-2.	TEX, S, C, and B Bit Field Encoding .....	75
2-3.	Cache Policy for Memory Attribute Encoding .....	75
2-4.	AP Bit Field Encoding .....	75
2-5.	Memory Region Attributes for MSP432P4xx Devices .....	76
2-6.	QNaN and SNaN Handling .....	80
2-7.	FPU Registers .....	83
2-8.	FPCCR Register Field Descriptions .....	84
2-9.	FPCAR Register Field Descriptions .....	85
2-10.	FPDSCR Register Field Descriptions .....	86
2-11.	MVFR0 Register Field Descriptions .....	87
2-12.	MVFR1 Register Field Descriptions .....	88
2-13.	MPU Registers .....	89
2-14.	TYPE Register Field Descriptions .....	90
2-15.	CTRL Register Field Descriptions .....	91
2-16.	RNR Register Field Descriptions .....	92
2-17.	RBAR Register Field Descriptions .....	93
2-18.	RASR Register Field Descriptions .....	94
2-19.	RBAR_A1 Register Field Descriptions .....	96
2-20.	RASR_A1 Register Field Descriptions .....	97
2-21.	RBAR_A2 Register Field Descriptions .....	99
2-22.	RASR_A2 Register Field Descriptions .....	100
2-23.	RBAR_A3 Register Field Descriptions .....	102
2-24.	RASR_A3 Register Field Descriptions .....	103
2-25.	NVIC Registers .....	105
2-26.	ISER0 Register Field Descriptions .....	106
2-27.	ISER1 Register Field Descriptions .....	106
2-28.	ICER0 Register Field Descriptions .....	107
2-29.	ICER1 Register Field Descriptions .....	107
2-30.	ISPR0 Register Field Descriptions .....	108
2-31.	ISPR1 Register Field Descriptions .....	108
2-32.	ICPR0 Register Field Descriptions .....	109
2-33.	ICPR1 Register Field Descriptions .....	109
2-34.	IABR0 Register Field Descriptions .....	110
2-35.	IABR1 Register Field Descriptions .....	110

2-36. IPR0 Register Field Descriptions .....	111
2-37. IPR1 Register Field Descriptions .....	111
2-38. IPR2 Register Field Descriptions .....	112
2-39. IPR3 Register Field Descriptions .....	112
2-40. IPR4 Register Field Descriptions .....	113
2-41. IPR5 Register Field Descriptions .....	113
2-42. IPR6 Register Field Descriptions .....	114
2-43. IPR7 Register Field Descriptions .....	114
2-44. IPR8 Register Field Descriptions .....	115
2-45. IPR9 Register Field Descriptions .....	115
2-46. IPR10 Register Field Descriptions .....	116
2-47. IPR11 Register Field Descriptions .....	116
2-48. IPR12 Register Field Descriptions .....	117
2-49. IPR13 Register Field Descriptions .....	117
2-50. IPR14 Register Field Descriptions .....	118
2-51. IPR15 Register Field Descriptions .....	118
2-52. STIR Register Field Descriptions .....	119
2-53. SYSTICK Registers .....	120
2-54. STCSR Register Field Descriptions .....	121
2-55. STRVR Register Field Descriptions .....	122
2-56. STCVR Register Field Descriptions .....	123
2-57. STCR Register Field Descriptions .....	124
2-58. SCB Registers .....	125
2-59. CPUID Register Field Descriptions .....	126
2-60. ICSR Register Field Descriptions .....	127
2-61. VTOR Register Field Descriptions .....	129
2-62. AIRCR Register Field Descriptions .....	130
2-63. SCR Register Field Descriptions .....	132
2-64. CCR Register Field Descriptions .....	133
2-65. SHPR1 Register Field Descriptions .....	134
2-66. SHPR2 Register Field Descriptions .....	135
2-67. SHPR3 Register Field Descriptions .....	136
2-68. SHCSR Register Field Descriptions .....	137
2-69. CFSR Register Field Descriptions .....	139
2-70. HFSR Register Field Descriptions .....	141
2-71. DFSR Register Field Descriptions .....	142
2-72. MMFAR Register Field Descriptions .....	143
2-73. BFAR Register Field Descriptions .....	144
2-74. AFSR Register Field Descriptions .....	145
2-75. PFR0 Register Field Descriptions .....	146
2-76. PFR1 Register Field Descriptions .....	147
2-77. DFR0 Register Field Descriptions .....	148
2-78. AFR0 Register Field Descriptions .....	149
2-79. MMFR0 Register Field Descriptions .....	150
2-80. MMFR1 Register Field Descriptions .....	151
2-81. MMFR2 Register Field Descriptions .....	152
2-82. MMFR3 Register Field Descriptions .....	153
2-83. ISAR0 Register Field Descriptions .....	154
2-84. ISAR1 Register Field Descriptions .....	155

---

2-85.	ISAR2 Register Field Descriptions .....	156
2-86.	ISAR3 Register Field Descriptions .....	157
2-87.	ISAR4 Register Field Descriptions .....	158
2-88.	CPACR Register Field Descriptions.....	159
2-89.	SCnSCB Registers .....	160
2-90.	ICTR Register Field Descriptions .....	161
2-91.	ACTLR Register Field Descriptions .....	162
2-92.	COREDEBUG Registers.....	163
2-93.	DHCSR Register Field Descriptions.....	164
2-94.	DCRSR Register Field Descriptions.....	166
2-95.	DCRDR Register Field Descriptions .....	167
2-96.	DEMCR Register Field Descriptions .....	168
2-97.	FPB Registers .....	170
2-98.	FP_CTRL Register Field Descriptions .....	171
2-99.	FP_REMAP Register Field Descriptions.....	172
2-100.	FP_COMP0 Register Field Descriptions.....	173
2-101.	FP_COMP1 Register Field Descriptions.....	174
2-102.	FP_COMP2 Register Field Descriptions.....	175
2-103.	FP_COMP3 Register Field Descriptions.....	176
2-104.	FP_COMP4 Register Field Descriptions.....	177
2-105.	FP_COMP5 Register Field Descriptions.....	178
2-106.	FP_COMP6 Register Field Descriptions.....	179
2-107.	FP_COMP7 Register Field Descriptions.....	180
2-108.	DWT Registers .....	181
2-109.	CTRL Register Field Descriptions .....	182
2-110.	CYCCNT Register Field Descriptions .....	184
2-111.	CPICNT Register Field Descriptions .....	185
2-112.	EXCCNT Register Field Descriptions .....	186
2-113.	SLEEPCNT Register Field Descriptions .....	187
2-114.	LSUCNT Register Field Descriptions .....	188
2-115.	FOLDCNT Register Field Descriptions .....	189
2-116.	PCSR Register Field Descriptions.....	190
2-117.	COMP0 Register Field Descriptions.....	191
2-118.	MASK0 Register Field Descriptions .....	192
2-119.	FUNCTION0 Register Field Descriptions.....	193
2-120.	COMP1 Register Field Descriptions.....	195
2-121.	MASK1 Register Field Descriptions .....	196
2-122.	FUNCTION1 Register Field Descriptions.....	197
2-123.	COMP2 Register Field Descriptions.....	199
2-124.	MASK2 Register Field Descriptions .....	200
2-125.	FUNCTION2 Register Field Descriptions.....	201
2-126.	COMP3 Register Field Descriptions.....	203
2-127.	MASK3 Register Field Descriptions .....	204
2-128.	FUNCTION3 Register Field Descriptions.....	205
2-129.	ITM Registers .....	207
2-130.	STIM0 Register Field Descriptions .....	208
2-131.	STIM1 Register Field Descriptions .....	209
2-132.	STIM2 Register Field Descriptions .....	210
2-133.	STIM3 Register Field Descriptions .....	211

2-134. STIM4 Register Field Descriptions .....	212
2-135. STIM5 Register Field Descriptions .....	213
2-136. STIM6 Register Field Descriptions .....	214
2-137. STIM7 Register Field Descriptions .....	215
2-138. STIM8 Register Field Descriptions .....	216
2-139. STIM9 Register Field Descriptions .....	217
2-140. STIM10 Register Field Descriptions.....	218
2-141. STIM11 Register Field Descriptions.....	219
2-142. STIM12 Register Field Descriptions.....	220
2-143. STIM13 Register Field Descriptions.....	221
2-144. STIM14 Register Field Descriptions.....	222
2-145. STIM15 Register Field Descriptions.....	223
2-146. STIM16 Register Field Descriptions.....	224
2-147. STIM17 Register Field Descriptions.....	225
2-148. STIM18 Register Field Descriptions.....	226
2-149. STIM19 Register Field Descriptions.....	227
2-150. STIM20 Register Field Descriptions.....	228
2-151. STIM21 Register Field Descriptions.....	229
2-152. STIM22 Register Field Descriptions.....	230
2-153. STIM23 Register Field Descriptions.....	231
2-154. STIM24 Register Field Descriptions.....	232
2-155. STIM25 Register Field Descriptions.....	233
2-156. STIM26 Register Field Descriptions.....	234
2-157. STIM27 Register Field Descriptions.....	235
2-158. STIM28 Register Field Descriptions.....	236
2-159. STIM29 Register Field Descriptions.....	237
2-160. STIM30 Register Field Descriptions.....	238
2-161. STIM31 Register Field Descriptions.....	239
2-162. TER Register Field Descriptions.....	240
2-163. TPR Register Field Descriptions.....	240
2-164. TCR Register Field Descriptions.....	241
2-165. IWR Register Field Descriptions .....	242
2-166. IMCR Register Field Descriptions .....	242
2-167. LAR Register Field Descriptions .....	243
2-168. LSR Register Field Descriptions .....	243
3-1. RSTCTL Registers.....	248
3-2. RSTCTL_RESET_REQ Register Description .....	249
3-3. RSTCTL_HARDRESET_STAT Register Description .....	250
3-4. RSTCTL_HARDRESET_CLR Register Description.....	251
3-5. RSTCTL_HARDRESET_SET Register Description.....	252
3-6. RSTCTL_SOFTRESET_STAT Register Description.....	253
3-7. RSTCTL_SOFTRESET_CLR Register Description .....	254
3-8. RSTCTL_SOFTRESET_SET Register Description .....	255
3-9. RSTCTL_PSSRESET_STAT Register Description .....	256
3-10. RSTCTL_PSSRESET_CLR Register Description .....	256
3-11. RSTCTL_PCMRESET_STAT Register Description .....	257
3-12. RSTCTL_PCMRESET_CLR Register Description .....	257
3-13. RSTCTL_PINRESET_STAT Register Description .....	258
3-14. RSTCTL_PINRESET_CLR Register Description .....	258

3-15.	RSTCTL_REBOOTRESET_STAT Register Description .....	259
3-16.	RSTCTL_REBOOTRESET_CLR Register Description .....	259
3-17.	RSTCTL_CSRESET_STAT Register Description .....	260
3-18.	RSTCTL_CSRESET_CLR Register Description .....	260
4-1.	Boot Override Flash Mailbox .....	271
4-2.	Commands Used by Boot-Code for Boot Override.....	276
4-3.	ACKs Used by Boot-Code to Indicate Status of Boot Override.....	276
4-4.	Tag Values.....	280
4-5.	Clock System Calibration Data.....	281
4-6.	ADC Calibration Data .....	282
4-7.	Flash Information Descriptor .....	283
4-8.	Random Number Seed.....	283
4-9.	BSL Configuration Data .....	283
4-10.	Structure of Device Identification Code .....	284
4-11.	SYSCTL Registers.....	285
4-12.	SYS_REBOOT_CTL Register Description .....	286
4-13.	SYS_NMI_CTLSTAT Register Description .....	287
4-14.	SYS_WDTRESET_CTL Register Description .....	288
4-15.	SYS_PERIHALT_CTL Register Description .....	289
4-16.	SYS_SRAM_SIZE Register Description .....	290
4-17.	SYS_SRAM_BANKEN Register Description .....	291
4-18.	SYS_SRAM_BANKRET Register Description .....	292
4-19.	SYS_FLASH_SIZE Register Description .....	293
4-20.	SYS_DIO_GLTFILT_CTL Register Description .....	294
4-21.	SYS_SECDATA_UNLOCK Register Description.....	295
4-22.	SYS_MASTER_UNLOCK Register Description .....	296
4-23.	SYS_BOOTOVER_REQ0 Register Description .....	297
4-24.	SYS_BOOTOVER_REQ1 Register Description .....	298
4-25.	SYS_BOOTOVER_ACK Register Description .....	299
4-26.	SYS_RESET_REQ Register Description .....	300
4-27.	SYS_RESET_STATOVER Register Description .....	301
4-28.	SYS_SYSTEM_STAT Register Description .....	302
5-1.	HXTXFREQ Settings .....	307
5-2.	CS Registers .....	318
5-3.	CSKEY Register Description .....	319
5-4.	CSCTL0 Register Description.....	320
5-5.	CSCTL1 Register Description.....	321
5-6.	CSCTL2 Register Description.....	323
5-7.	CSCTL3 Register Description.....	325
5-8.	CSCLKEN Register Description .....	326
5-9.	CSSTAT Register Description .....	327
5-10.	CSIE Register Description .....	329
5-11.	CSIFG Register Description .....	330
5-12.	CSCLRIFG Register Description .....	331
5-13.	CSSETIFG Register Description.....	332
5-14.	CSDCOERCAL0 Register Description .....	333
5-15.	CSDCOERCAL1 Register Description .....	334
6-1.	PSS Registers.....	340
6-2.	PSSKEY Register Description .....	341

6-3.	PSSCTL0 Register Description .....	342
6-4.	PSSIE Register Description .....	344
6-5.	PSSIFG Register Description .....	345
6-6.	PSSCLRIFG Register Description .....	346
7-1.	Power Modes Summary .....	352
7-2.	Power Mode Selection .....	358
7-3.	AM Invalid Transition NMI/interrupt Enable .....	359
7-4.	LPM Invalid Transition NMI/Interrupt Enable .....	359
7-5.	LPM Clock Checks NMI/Interrupt Enable .....	360
7-6.	DC-DC Error NMI/interrupt Enable .....	361
7-7.	SVSMH Performance and Power Modes .....	365
7-8.	Wake-up Sources from Low Power Modes .....	367
7-9.	PCM Registers .....	368
7-10.	PCMCTL0 Register Description .....	369
7-11.	PCMCTL1 Register Description .....	371
7-12.	PCMIE Register Description .....	373
7-13.	PCMIFG Register Description .....	374
7-14.	PCMCLRIFG Register Description .....	375
8-1.	MSP432 Driver Library API for Flash Wait-State Configuration .....	378
8-2.	MSP432 Driver Library API for Flash Read Buffering Configuration .....	379
8-3.	MSP432 Driver Library API for Flash Program Operation .....	379
8-4.	MSP432 Driver Library API for Flash Sector Erase Operation .....	379
8-5.	MSP432 Driver Library API for Flash Mass Erase Operation .....	380
8-6.	MSP432 Driver Library API for Setting up Program or Erase Operation .....	380
8-7.	MSP432 Driver Library API for Setting up Flash Read Modes .....	381
8-8.	Configuring the Auto-Verify Mode Through Direct Register Access .....	382
8-9.	MSP432 Driver Library API for Setting up Auto-Verify Before Program Operations .....	382
8-10.	MSP432 Driver Library API for Enabling Program Operations .....	383
8-11.	MSP432 Driver Library API for Flash Erase Operations .....	391
8-12.	FLCTL Registers .....	395
8-13.	FLCTL_POWER_STAT Register Description .....	397
8-14.	FLCTL_BANK0_RDCTL Register Description .....	398
8-15.	FLCTL_BANK1_RDCTL Register Description .....	400
8-16.	FLCTL_RDBRST_CTLSTAT Register Description .....	402
8-17.	FLCTL_RDBRST_STARTADDR Register Description .....	403
8-18.	FLCTL_RDBRST_LEN Register Description .....	404
8-19.	FLCTL_RDBRST_FAILADDR Register Description .....	405
8-20.	FLCTL_RDBRST_FAILCNT Register Description .....	406
8-21.	FLCTL_PRG_CTLSTAT Register Description .....	407
8-22.	FLCTL_PRGBRST_CTLSTAT Register Description .....	408
8-23.	FLCTL_PRGBRST_STARTADDR Register Description .....	410
8-24.	FLCTL_PRGBRST_DATA0_0 Register Description .....	411
8-25.	FLCTL_PRGBRST_DATA0_1 Register Description .....	411
8-26.	FLCTL_PRGBRST_DATA0_2 Register Description .....	412
8-27.	FLCTL_PRGBRST_DATA0_3 Register Description .....	412
8-28.	FLCTL_PRGBRST_DATA1_0 Register Description .....	413
8-29.	FLCTL_PRGBRST_DATA1_1 Register Description .....	413
8-30.	FLCTL_PRGBRST_DATA1_2 Register Description .....	414
8-31.	FLCTL_PRGBRST_DATA1_3 Register Description .....	414

8-32.	FLCTL_PRGBRST_DATA2_0 Register Description .....	415
8-33.	FLCTL_PRGBRST_DATA2_1 Register Description .....	415
8-34.	FLCTL_PRGBRST_DATA2_2 Register Description .....	416
8-35.	FLCTL_PRGBRST_DATA2_3 Register Description .....	416
8-36.	FLCTL_PRGBRST_DATA3_0 Register Description .....	417
8-37.	FLCTL_PRGBRST_DATA3_1 Register Description .....	417
8-38.	FLCTL_PRGBRST_DATA3_2 Register Description .....	418
8-39.	FLCTL_PRGBRST_DATA3_3 Register Description .....	418
8-40.	FLCTL_ERASE_CTLSTAT Register Description.....	419
8-41.	FLCTL_ERASE_SECTADDR Register Description .....	420
8-42.	FLCTL_BANK0_INFO_WEPROT Register Description .....	421
8-43.	FLCTL_BANK0_MAIN_WEPROT Register Description .....	422
8-44.	FLCTL_BANK1_INFO_WEPROT Register Description .....	424
8-45.	FLCTL_BANK1_MAIN_WEPROT Register Description .....	425
8-46.	FLCTL_BMRK_CTLSTAT Register Description .....	427
8-47.	FLCTL_BMRK_IFETCH Register Description .....	428
8-48.	FLCTL_BMRK_DREAD Register Description.....	429
8-49.	FLCTL_BMRK_CMP Register Description .....	430
8-50.	FLCTL_IFG Register Description .....	431
8-51.	FLCTL_IE Register Description.....	432
8-52.	FLCTL_CLRIFG Register Description .....	433
8-53.	FLCTL_SETIFG Register Description.....	434
8-54.	FLCTL_READ_TIMCTL Register Description.....	435
8-55.	FLCTL_READMARGIN_TIMCTL Register Description .....	436
8-56.	FLCTL_PRGVER_TIMCTL Register Description.....	437
8-57.	FLCTL_ERSVER_TIMCTL Register Description .....	438
8-58.	FLCTL_PROGRAM_TIMCTL Register Description .....	439
8-59.	FLCTL_ERASE_TIMCTL Register Description .....	440
8-60.	FLCTL_MASSERASE_TIMCTL Register Description .....	441
8-61.	FLCTL_BURSTPRG_TIMCTL Register Description .....	442
9-1.	Protection Signaling .....	446
9-2.	Address Increments .....	446
9-3.	Key Handshake Rules for the DMA Controller.....	447
9-4.	AHB Bus Transfer Arbitration Interval.....	449
9-5.	DMA Channel Priority .....	450
9-6.	DMA Cycle Types.....	451
9-7.	channel_cfg for a Primary Data Structure, In Memory Scatter-Gather Mode.....	455
9-8.	channel_cfg for a Primary Data Structure, in Peripheral Scatter-Gather Mode .....	458
9-9.	Address Bit Settings for the Channel Control Data Structure .....	462
9-10.	Permitted Base Addresses .....	463
9-11.	rc_data_end_ptr Bit Assignments .....	464
9-12.	dst_data_end_ptr Bit Assignments .....	464
9-13.	channel_cfg Bit Assignments.....	465
9-14.	DMA Registers .....	469
9-15.	DMA_DEVICE_CFG Register Description .....	470
9-16.	DMA_SW_CHTRIG Register Description .....	471
9-17.	DMA_CHn_SRCCFG Register Description .....	473
9-18.	DMA_INT1_SRCCFG Register Description .....	474
9-19.	DMA_INT2_SRCCFG Register Description .....	475

9-20.	DMA_INT3_SRCCFG Register Description .....	476
9-21.	DMA_INT0_SRCFLG Register Description .....	477
9-22.	DMA_INT0_CLRFLG Register Description.....	478
9-23.	DMA_STAT Register Field Descriptions .....	480
9-24.	DMA_CFG Register Field Descriptions .....	481
9-25.	DMA_CTLBASE Register Field Descriptions .....	482
9-26.	DMA_ALTBASE Register Field Descriptions .....	483
9-27.	DMA_WAITSTAT Register Field Descriptions .....	484
9-28.	DMA_SWREQ Register Field Descriptions .....	485
9-29.	DMA_USEBURSTSET Register Field Descriptions.....	486
9-30.	DMA_USEBURSTCLR Register Field Descriptions.....	487
9-31.	DMA_REQMASKSET Register Field Descriptions .....	488
9-32.	DMA_REQMASKCLR Register Field Descriptions.....	489
9-33.	DMA_ENASET Register Field Descriptions .....	490
9-34.	DMA_ENACLAR Register Field Descriptions.....	491
9-35.	DMA_ALTSET Register Field Descriptions .....	492
9-36.	DMA_ALTCLEAR Register Field Descriptions .....	493
9-37.	DMA_PRIOSSET Register Field Descriptions .....	494
9-38.	DMA_PRIOCLEAR Register Field Descriptions.....	495
9-39.	DMA_ERRCLEAR Register Field Descriptions.....	496
10-1.	I/O Configuration .....	499
10-2.	I/O Function Selection.....	500
10-3.	Digital I/O Registers .....	504
10-4.	PxIV Register Description .....	516
10-5.	PxIN Register Description .....	517
10-6.	PxOUT Register Description .....	517
10-7.	PxDIR Register Description .....	517
10-8.	PxREN Register Description .....	518
10-9.	PxDS Register Description .....	518
10-10.	PxSEL0 Register Description .....	518
10-11.	PxSEL1 Register Description .....	519
10-12.	PxSELC Register Description .....	519
10-13.	P1IES Register Description .....	519
10-14.	PxIE Register Description .....	520
10-15.	PxIFG Register Description .....	520
11-1.	Examples for Port Mapping Mnemonics and Functions .....	523
11-2.	PMAP Registers .....	525
11-3.	PMAPKEYID Register Description .....	527
11-4.	PMAPCTL Register Description .....	527
11-5.	P1MAP0 to P1MAP7 Register Description .....	527
11-6.	P2MAP0 to P2MAP7 Register Description .....	528
11-7.	P3MAP0 to P3MAP7 Register Description .....	528
11-8.	P4MAP0 to P4MAP7 Register Description .....	528
11-9.	P5MAP0 to P5MAP7 Register Description .....	529
11-10.	P6MAP0 to P6MAP7 Register Description .....	529
11-11.	P7MAP0 to P7MAP7 Register Description .....	529
11-12.	PxMAPy Register Description .....	529
12-1.	CapTouch Registers.....	533
12-2.	CAPTOIOxCTL Register Description.....	534

13-1.	CRC32 Registers .....	538
13-2.	CRC32DI Register Description.....	539
13-3.	CRC32DIRB Register Description .....	540
13-4.	CRC32INIRES_LO Register Description .....	541
13-5.	CRC32INIRES_HI Register Description .....	542
13-6.	CRC32RESR_LO Register Description.....	543
13-7.	CRC32RESR_HI Register Description.....	544
13-8.	CRC16DI Register Description.....	545
13-9.	CRC16DIRB Register Description.....	546
13-10.	CRC16INIRES Register Description .....	547
13-11.	CRC16RESR Register Description .....	548
14-1.	AES Operation Modes Overview .....	551
14-2.	'AES trigger 0-2' Operation When AESCMEN = 1 .....	557
14-3.	AES and DMA Configuration for ECB Encryption .....	559
14-4.	AES DMA Configuration for ECB Decryption .....	560
14-5.	AES and DMA Configuration for CBC Encryption .....	561
14-6.	AES and DMA Configuration for CBC Decryption .....	562
14-7.	AES and DMA Configuration for OFB Encryption .....	563
14-8.	AES and DMA Configuration for OFB Decryption .....	564
14-9.	AES and DMA Configuration for CFB Encryption .....	565
14-10.	AES and DMA Configuration for CFB Decryption .....	566
14-11.	AES256 Registers .....	567
14-12.	AESACTL0 Register Description .....	568
14-13.	AESACTL1 Register Description .....	570
14-14.	AESASTAT Register Description .....	571
14-15.	AESAKEY Register Description.....	572
14-16.	AESADIN Register Description .....	573
14-17.	AESADOUT Register Description .....	574
14-18.	AESAXDIN Register Description .....	575
14-19.	AESAXIN Register Description .....	576
15-1.	WDT_A Clock Sources.....	581
15-2.	WDT_A Registers.....	583
15-3.	WDTCTL Register Description .....	584
16-1.	Timer32 Registers .....	588
16-2.	T32LOAD1 Register Description.....	589
16-3.	T32VALUE1 Register Description .....	590
16-4.	T32CONTROL1 Register Description .....	591
16-5.	T32INTCLR1 Register Description .....	592
16-6.	T32RIS1 Register Description .....	593
16-7.	T32MIS1 Register Description .....	594
16-8.	T32BGLOAD1 Register Description.....	595
16-9.	T32LOAD2 Register Description.....	596
16-10.	T32VALUE2 Register Description .....	597
16-11.	T32CONTROL2 Register Description .....	598
16-12.	T32INTCLR2 Register Description .....	599
16-13.	T32RIS2 Register Description .....	600
16-14.	T32MIS2 Register Description .....	601
16-15.	T32BGLOAD2 Register Description.....	602
17-1.	Timer Modes .....	607

17-2.	Output Modes .....	612
17-3.	Timer_A Registers .....	617
17-4.	TAxCTL Register Description .....	618
17-5.	TAxR Register Description .....	619
17-6.	TAxCCTL0 to TAxCCTL6 Register Description .....	620
17-7.	TAxCCR0 to TAxCCR6 Register Description .....	622
17-8.	TAxIV Register Description .....	622
17-9.	TAxEX0 Register Description .....	623
18-1.	RTC_C Registers .....	634
18-2.	RTCCTL0_L Register Description .....	636
18-3.	RTCCTL0_H Register Description .....	637
18-4.	RTCCTL1 Register Description .....	638
18-5.	RTCCTL3 Register Description .....	639
18-6.	RTCOCAL Register Description .....	640
18-7.	RTCTCMP Register Description .....	641
18-8.	RTCSEC Register Description .....	642
18-9.	RTCSEC Register Description .....	642
18-10.	RTCMIN Register Description .....	643
18-11.	RTCMIN Register Description .....	643
18-12.	RTCHOUR Register Description .....	644
18-13.	RTCHOUR Register Description .....	644
18-14.	RTCDOW Register Description .....	645
18-15.	RTCDAY Register Description .....	645
18-16.	RTCDAY Register Description .....	645
18-17.	RTCMON Register Description .....	646
18-18.	RTCMON Register Description .....	646
18-19.	RTCYEAR Register Description .....	647
18-20.	RTCYEAR Register Description .....	647
18-21.	RTCAMIN Register Description .....	648
18-22.	RTCAMIN Register Description .....	648
18-23.	RTCAHOUR Register Description .....	649
18-24.	RTCAHOUR Register Description .....	649
18-25.	RTCADOW Register Description .....	650
18-26.	RTCADAY Register Description .....	650
18-27.	RTCADAY Register Description .....	650
18-28.	RTCPSP0CTL Register Description .....	651
18-29.	RTCPSP1CTL Register Description .....	652
18-30.	RTCPSP0 Register Description .....	653
18-31.	RTCPSP1 Register Description .....	653
18-32.	RTCIV Register Description .....	654
18-33.	RTCBIN2BCD Register Description .....	655
18-34.	RTCBIN2BIN Register Description .....	655
19-1.	REF_A Registers .....	660
19-2.	REFCTL0 Register Description .....	661
20-1.	ADC14 Conversion Result Formats .....	670
20-2.	Conversion Mode Summary .....	671
20-3.	ADC14 Registers .....	680
20-4.	ADC14CTL0 Register Description .....	681
20-5.	ADC14CTL1 Register Description .....	684

20-6.	ADC14LO0 Register Description .....	686
20-7.	ADC14HI0 Register Description .....	687
20-8.	ADC14LO1 Register Description .....	688
20-9.	ADC14HI1 Register Description .....	689
20-10.	ADC14MCTL0 to ADC14MCTL31 Register Description .....	690
20-11.	ADC14MEM0 to ADC14MEM31 Register Description .....	692
20-12.	ADC14IER0 Register Description .....	693
20-13.	ADC14IER1 Register Description .....	696
20-14.	ADC14IFGR0 Register Description .....	697
20-15.	ADC14IFGR1 Register Description .....	701
20-16.	ADC14CLRIFGR0 Register Description .....	702
20-17.	ADC14CLRIFGR1 Register Description .....	705
20-18.	ADC14IV Register Description .....	707
21-1.	COMP_E Registers.....	716
21-2.	CExCTL0 Register Description .....	717
21-3.	CExCTL1 Register Description .....	718
21-4.	CExCTL2 Register Description .....	719
21-5.	CExCTL3 Register Description .....	720
21-6.	CExINT Register Description.....	722
21-7.	CExIV Register Description .....	723
22-1.	Receive Error Conditions .....	732
22-2.	Modulation Pattern Examples .....	734
22-3.	BITCLK16 Modulation Pattern .....	735
22-4.	UCBRSx Settings for Fractional Portion of N = $f_{BRCLK}/\text{Baud Rate}$ .....	736
22-5.	Recommended Settings for Typical Crystals and Baud Rates .....	739
22-6.	UART State Change Interrupt Flags .....	741
22-7.	eUSCI_A UART Registers.....	743
22-8.	UCAxCTLW0 Register Description .....	744
22-9.	UCAxCTLW1 Register Description .....	745
22-10.	UCAxBRW Register Description.....	746
22-11.	UCAxMCTLW Register Description .....	746
22-12.	UCAxSTATW Register Description .....	747
22-13.	UCAxRXBUF Register Description .....	748
22-14.	UCAxTXBUF Register Description .....	748
22-15.	UCAxABCTL Register Description .....	749
22-16.	UCAxIRCTL Register Description .....	750
22-17.	UCAxIE Register Description.....	751
22-18.	UCAxIFG Register Description.....	752
22-19.	UCAxIV Register Description.....	753
23-1.	UCxSTE Operation .....	757
23-2.	eUSCI_A SPI Registers.....	763
23-3.	UCAxCTLW0 Register Description .....	764
23-4.	UCAxBRW Register Description .....	765
23-5.	UCAxSTATW Register Description .....	766
23-6.	UCAxRXBUF Register Description .....	767
23-7.	UCAxTXBUF Register Description .....	768
23-8.	UCAxIE Register Description.....	769
23-9.	UCAxIFG Register Description.....	770
23-10.	UCAxIV Register Description.....	771

---

23-11. eUSCI_B SPI Registers.....	772
23-12. UCBxCTLW0 Register Description .....	773
23-13. UCBxBRW Register Description .....	774
23-14. UCBxSTATW Register Description .....	774
23-15. UCBxRXBUF Register Description .....	775
23-16. UCBxTXBUF Register Description .....	775
23-17. UCBxIE Register Description.....	776
23-18. UCBxIFG Register Description.....	776
23-19. UCBxIV Register Description.....	777
24-1. Glitch Filter Length Selection Bits .....	793
24-2. I <sup>2</sup> C State Change Interrupt Flags .....	798
24-3. eUSCI_B Registers .....	799
24-4. UCBxCTLW0 Register Description .....	800
24-5. UCBxCTLW1 Register Description .....	802
24-6. UCBxBRW Register Description .....	804
24-7. UCBxSTATW Register Description .....	804
24-8. UCBxTBCNT Register Description .....	805
24-9. UCBxRXBUF Register Description .....	806
24-10. UCBxTXBUF Register Description .....	806
24-11. UCBxI2COA0 Register Description .....	807
24-12. UCBxI2COA1 Register Description .....	808
24-13. UCBxI2COA2 Register Description .....	808
24-14. UCBxI2COA3 Register Description .....	809
24-15. UCBxADDRX Register Description.....	809
24-16. UCBxADDMASK Register Description.....	810
24-17. UCBxI2CSA Register Description .....	810
24-18. UCBxIE Register Description.....	811
24-19. UCBxIFG Register Description.....	813
24-20. UCBxIV Register Description.....	815

## Read This First

---



---



---

### **Register Bit Conventions, Accessibility, and Initial Condition**

Key	Bit Accessibility
rw	Read/write
r	Read only. If there is no reset value assigned, always reads as 0. Writes have no effect.
w	Write only. Always reads as 0.
-{0},-{1}	Condition after Soft Reset (or higher class reset)
-0,-1	Condition after Hard Reset (or higher class reset)
-<0>,-<1>	Condition after Reboot Reset (or higher class reset)
-(0),-(1)	Condition after POR

Refer to the Reset section in the device-specific data sheet and the Reset Controller chapter in this Technical Reference Manual for more details on the reset hierarchy in each device.

### **Bit Definition Examples**

Example	Remarks
rw-0	Bit that can be read or written. It is reset to a value of 0h on a Hard Reset.
rw-(0)	Bit that can be read or written. It is reset to a value of 0h on a POR.
r-{1}	Read only bit. It is set to a value of 1h on a Soft Reset.
r	Read only bit. Always reads as 0h. A reserved bit.
w	Write only bit. Always reads as 0h.

## Cortex-M4F Processor

The ARM® Cortex®-M4F processor provides a high-performance low-cost platform that meets the system requirements of minimal memory implementation, reduced pin count, and low power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

Topic	Page
1.1 Introduction .....	41
1.2 Overview .....	44
1.3 Programming Model .....	46
1.4 Memory Model .....	50
1.5 Exception Model .....	55
1.6 Fault Handling .....	62
1.7 Power Management .....	64
1.8 Instruction Set Summary.....	65

## 1.1 Introduction

The Cortex-M4F implementation in MSP432P4xx incorporates the following:

- ARM Cortex-M4 processor core (Revision r0p1) based on ARMV7-M architecture
- Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low-latency interrupt processing; supports up to 64 interrupt sources
- Multiple high-performance bus interfaces
- Low-cost debug solution with the ability to:
  - Implement breakpoints through FPB
  - Implement watchpoints, tracing, and system profiling through DWT
- Memory Protection Unit (MPU) supports eight regions
- IEEE 754-compliant Floating Point Unit (FPU) unit for fast floating point processing
- Bit-banding support for SRAM and peripherals
- SysTICK timer for periodic ticks

### The Cortex-M4 processor features:

- Low-gate-count processor core, with low-latency interrupt processing that has:
  - A subset of the Thumb instruction set, defined in the [ARMv7-M Architecture Reference Manual](#).
  - Thumb-2 mixed 16- and 32-bit instruction set delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices, for microcontroller-class applications
    - Single-cycle multiply instruction and hardware divide
    - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
    - Unaligned data access, enabling data to be efficiently packed into memory
  - IEEE 754-compliant single-precision Floating-Point Unit (FPU)
  - 16-bit SIMD vector processing unit
  - Banked stack pointer (SP)
  - Hardware integer divide instructions, SDIV and UDIV
  - Handler and Thread modes of processor operation
  - Thumb and Debug states
  - Support for interruptible-continued instructions LDM, STM, PUSH, and POP for low interrupt latency
  - Automatic processor state saving and restoration for low latency Interrupt Service Routine (ISR) entry and exit
  - Support for ARM unaligned accesses
- Floating Point Unit (FPU) in the Cortex-M4F processor provides:
  - 32-bit instructions for single-precision (C float) data-processing operations
  - Combined multiply-and-accumulate instructions for increased precision (Fused MAC).
  - Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root
  - Hardware support for denormals and all IEEE rounding modes
  - 32 dedicated 32-bit single precision registers, also addressable as 16 double-word registers
  - Decoupled three stage pipeline
- Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low-latency interrupt processing. Features include:
  - Support for up to 64 interrupt sources
  - Three bits to define the priority of each interrupt (total of eight priority levels)
  - Dynamic reprioritization of interrupts

- Priority grouping. This enables selection of preempting interrupt levels and non preempting interrupt levels.
- Support for tail-chaining and late arrival of interrupts. This enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.
- Processor state automatically saved on interrupt entry, and restored on interrupt exit, with no instruction overhead.
- Memory Protection Unit (MPU) includes:
  - Eight memory regions
  - Sub Region Disable (SRD), enabling efficient use of memory regions
  - The ability to enable a background region that implements the default memory map attributes
- Bus interfaces:
  - Three Advanced High-performance Bus-Lite (AHB-Lite) interfaces: ICode, DCode, and System bus interfaces
  - Private Peripheral Bus (PPB) based on Advanced Peripheral Bus (APB) interface
  - Bit-band support that includes atomic bit-band write and read operations
  - Memory access alignment
  - Write buffer for buffering of write data
- Low-cost debug solution that features:
  - Debug access to all memory and registers in the system, including access to memory mapped devices, access to internal core registers when the core is halted, and access to debug control registers even while SYSRESETn is asserted
  - Support for Serial Wire and JTAG based debug ports
  - Flash Patch and Breakpoint (FPB) unit for implementing hardware breakpoints
  - Data Watchpoint and Trace (DWT) unit for implementing watchpoints, data tracing, and system profiling
  - Instrumentation Trace Macrocell (ITM) for support of printf() style debugging
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer (TPA), including Single Wire Output (SWO) mode

**Table 1-1. Cortex-M4F Optional Parameters Configuration in MSP432P4xx**

SL NO	Configuration Feature	Cortex M4 Options	MSP432P4xx Configuration
1	Number of user Interrupts	1-240	64
2	Levels of Interrupt Priority	8-256 Levels	8 Levels
3	Memory Protection Unit	Present or Absent	Present
4	Floating Point Unit	Present or Absent	Present
5	Bit Banding Support	Present or Absent	Present
6	Endianness	Big Endian or Little Endian	Little Endian
7	Wakeup Interrupt Controller	Present or Absent	Absent (instead a MSP432P4xx device-level wakeup controller is present)
8	Reset All Registers	Only Reset architecturally required registers or Reset all registers	Reset all registers
9	SYSTICK Calibration	Predefined calibration values for 10-ms count. Present or Absent	Absent. No support for the calibration, because the value depends on the operating frequency of the device, which is user specific.
10	Jtag Debug Port	Present or Absent	Present
11	Debug support level	No Debug, Minimal Debug, Full debug without data matching, Full debug with Data matching	Full debug with data matching. Debug port, AHB-AP, FPB and DWT present

**Table 1-1. Cortex-M4F Optional Parameters Configuration in MSP432P4xx (continued)**

SL NO	Configuration Feature	Cortex M4 Options	MSP432P4xx Configuration
12	Trace support level	No Trace, Standard Trace, Full Trace, Full Trace plus HTM port	Standard trace: ITM, TPIU and DWT triggers and counters present. ETM and HTM are not present.

This chapter provides information on the implementation of the Cortex-M4F processor, including the programming model, the memory model, the exception model and fault handling.

For technical details on the instruction set, see the Cortex-M4 instruction set chapter in the [ARM® Cortex-M4 Devices Generic User Guide](#).

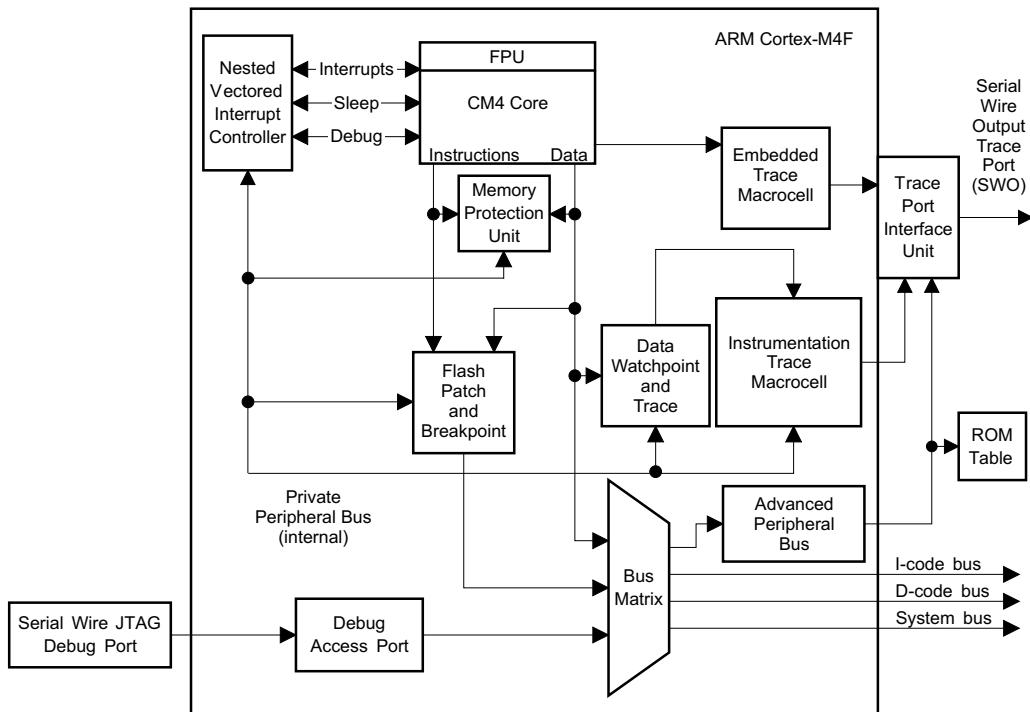
### 1.1.1 Block Diagram

The Cortex-M4F processor is built on a high-performance processor core with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including IEEE754-compliant single-precision floating-point computation, a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M4F processor implements tightly coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M4F processor implements a version of the Thumb® instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements. The Cortex-M4F instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The Cortex-M4F processor closely integrates a nested interrupt controller (NVIC), to deliver industry-leading interrupt performance. The MSP432P4xx NVIC includes a nonmaskable interrupt (NMI) and provides eight interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing interrupt latency. The hardware stacking of registers and the ability to suspend load-multiple and store-multiple operations further reduce interrupt latency. Interrupt handlers do not require any assembler stubs, which removes code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another. To optimize low-power designs, the NVIC integrates with the sleep modes, including Deep-sleep mode, which enables the entire device to be rapidly powered down.

[Figure 1-1](#) shows the CPU block diagram.



Copyright © 2016, Texas Instruments Incorporated

**Figure 1-1. CPU Block Diagram**

## 1.2 Overview

### 1.2.1 Bus Interface

MSP432P4xx Cortex-M4F implementation contains three high-speed AMBA® technology AHB-Lite Bus interfaces named ICODE, DCODE, and SBUS (System Bus) and one AMBA technology APB bus named PPB (Private Peripheral Bus). ICODE and DCODE buses support Harvard implementation of the processor, allowing separate paths for Instruction (ICODE) and Data (DCODE) accesses during code execution. [Table 1-2](#) lists the slaves mapped on each of these buses.

**NOTE:** The exact address range for each of the bus interfaces can be found in the device-specific data sheet. The values shown in [Table 1-2](#) are examples only.

**Table 1-2. Cortex-M4F Bus Interfaces in MSP432P4xx**

Sl. No.	Bus Interface Name	Protocol Type	Valid Address Range (Indicative Only)	Description
1	ICODE	AHB-Lite	0x0000_0000–0x1FFF_FFFF	Used for Instruction Access within the address range. Connects to Flash, ROM and SRAM.
2	DCODE	AHB-Lite	0x0000_0000–0x1FFF_FFFF	Used for Data Access within the address range. Connects to Flash, ROM and SRAM.
3	SBUS	AHB-Lite	0x2000_0000–0xDFFF_FFFF	Used for Data access within the address range. Connects to SRAM <sup>(1)</sup> and on-chip peripherals.
4	PPB	APB (v3.0)	0xE004_0000–0xE00F_FFFF	Connects to some system-critical modules like RSTCTL and SYSCTL. <sup>(2)</sup> Also the Core internal components like NVIC and MPU are mapped on this bus.

<sup>(1)</sup> SRAM mapped on SBUS should not be used for code execution, but only for stack and data storage.

<sup>(2)</sup> Mapping RSTCTL and SYSCTL on PPB allows accesses to the registers even if device is locked under Hard Reset. This is useful for debugging some of the lockup conditions.

### 1.2.2 Integrated Configurable Debug

The Cortex-M4F processor implements a complete hardware debug solution, providing high system visibility of the processor and memory through either a traditional JTAG port or a 2-pin Serial Wire Debug (SWD) port that is ideal for microcontrollers and other small package devices.

For system trace, the processor integrates an Instrumentation Trace Macrocell (ITM) alongside data watchpoints and a profiling unit. To enable simple and cost-effective profiling of the system trace events, a Serial Wire Viewer (SWV) can export a stream of software-generated messages, data trace, and profiling information through a single pin.

The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use.

A TPIU (Trace Port Interface Unit) acts as a bridge between the Cortex-M4F trace data from the ITM, and an off-chip Trace analyzer.

For more information on each of these blocks, refer to [Section 2.3](#) in Cortex-M4 Peripherals.

### 1.2.3 Cortex-M4F System Component Details

The Cortex-M4F includes the following system components:

- SysTick: A 24-bit count-down timer that can be used as a Real-Time Operating System (RTOS) tick timer or as a simple counter (see [Section 2.2.1](#)).
- Nested Vectored Interrupt Controller (NVIC): An embedded interrupt controller that supports low latency interrupt processing (see [Section 2.2.2](#)).
- System Control Block (SCB): The programming model interface to the processor. The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions (see [Section 2.2.3](#)).
- Memory Protection Unit (MPU): Improves system reliability by defining the memory attributes for different memory regions. The MPU provides up to eight different regions and an optional predefined background region (see [Section 2.2.4](#)).
- Floating-Point Unit (FPU): Fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square-root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions (see [Section 2.2.5](#)).

## 1.3 Programming Model

This section describes the Cortex-M4F programming model. In addition to the individual core register descriptions, information about the processor modes and privilege levels for software execution and stacks is included.

### 1.3.1 Processor Mode and Privilege Levels for Software Execution

The Cortex-M4F has two modes of operation:

- Thread mode: Used to execute application software. The processor enters Thread mode when it comes out of reset.
- Handler mode: Used to handle exceptions. When the processor has finished exception processing, it returns to Thread mode.

In addition, the Cortex-M4F has two privilege levels:

- Unprivileged: In this mode, software has the following restrictions:
  - Limited access to the MSR and MRS instructions and no use of the CPS instruction
  - No access to the system timer, NVIC, or system control block
  - Possibly restricted access to memory or peripherals
- Privileged: In this mode, software can use all the instructions and has access to all resources.

In Thread mode, the CONTROL register controls whether software execution is privileged or unprivileged. In Handler mode, software execution is always privileged.

Only privileged software can write to the CONTROL register to change the privilege level for software execution in Thread mode. Unprivileged software can use the SVC instruction to make a supervisor call to transfer control to privileged software.

### 1.3.2 Stacks

The processor uses a full descending stack, meaning that the stack pointer indicates the last stacked item on the memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks: the main stack and the process stack, with a pointer for each held in independent registers (see the SP register in [Section 1.3.4.2](#)).

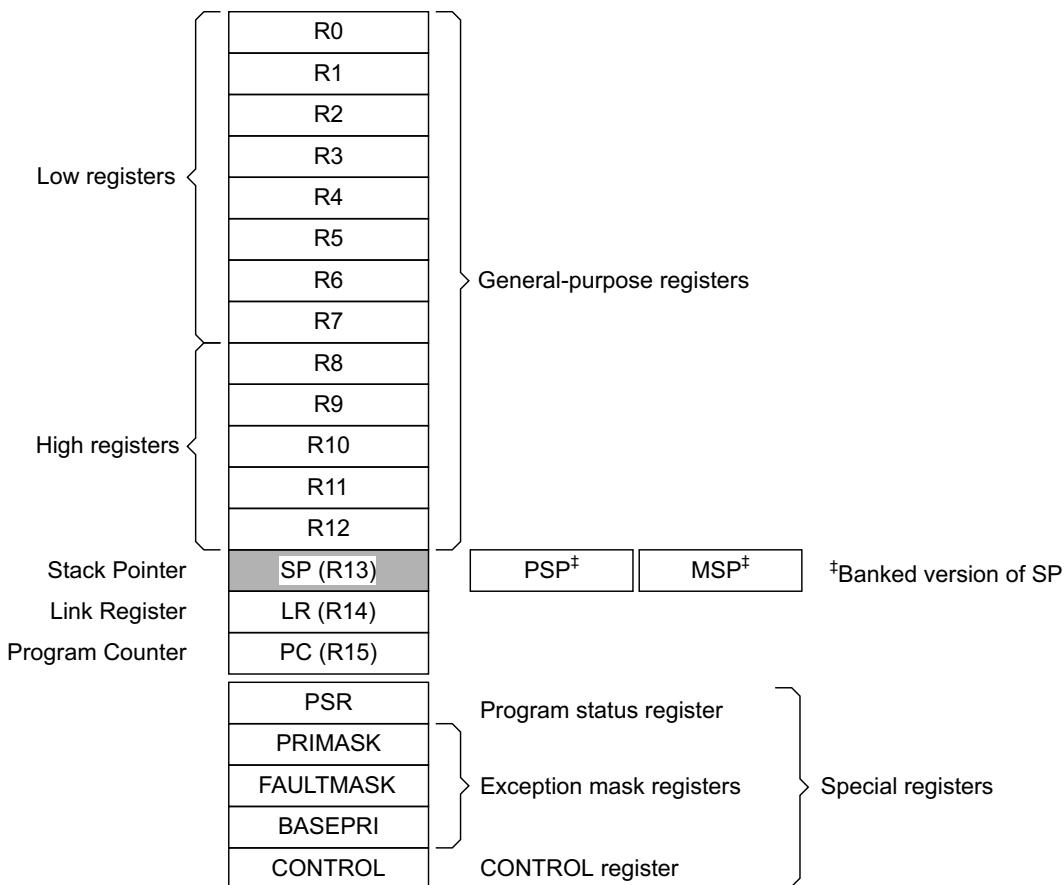
In Thread mode, the CONTROL register controls whether the processor uses the main stack or the process stack. In Handler mode, the processor always uses the main stack. The options for processor operations are shown in [Table 1-3](#).

**Table 1-3. Summary of Processor Mode, Privilege Level, and Stack Use**

Processor Mode	Use	Privilege Level	Stack Used
Thread	Applications	Privileged or unprivileged	Main stack or process stack
Handler	Exception handlers	Always privileged	Main stack

### 1.3.3 Register Map

Figure 1-2 shows the Cortex-M4F register set. The core registers are not memory mapped and are accessed by register name, so the base address is n/a (not applicable) and there is no offset.



**Figure 1-2. Cortex-M4F Register Set**

### 1.3.4 Register Descriptions

This section lists and describes the Cortex-M4F registers. The core registers are not memory mapped and are accessed by register name rather than offset.

#### 1.3.4.1 Register n: Cortex General Purpose Registers

- Register 0: Cortex General-Purpose Register 0 (R0)
- Register 1: Cortex General-Purpose Register 1 (R1)
- Register 2: Cortex General-Purpose Register 2 (R2)
- Register 3: Cortex General-Purpose Register 3 (R3)
- Register 4: Cortex General-Purpose Register 4 (R4)
- Register 5: Cortex General-Purpose Register 5 (R5)
- Register 6: Cortex General-Purpose Register 6 (R6)
- Register 7: Cortex General-Purpose Register 7 (R7)
- Register 8: Cortex General-Purpose Register 8 (R8)
- Register 9: Cortex General-Purpose Register 9 (R9)

Register 10: Cortex General-Purpose Register 10 (R10)

Register 11: Cortex General-Purpose Register 11 (R11)

Register 12: Cortex General-Purpose Register 12 (R12)

The Rn (R0 to R12) registers are 32-bit general-purpose registers for data operations and can be accessed from either privileged or unprivileged mode. They have no special architecturally defined uses. Most instructions that can specify a general-purpose register can specify R0 to R12.

#### Low Registers

- Registers R0 to R7 are accessible by all instructions that specify a general-purpose register.

#### High Registers

- Registers R8 to R12 are accessible by all 32-bit instructions that specify a general-purpose register.
- Registers R8 to R12 are not accessible by any 16-bit instructions.
- Registers R13, R14, and R15 have the following special functions.

#### 1.3.4.2 Register 13: Stack Pointer (SP, R13)

The Stack Pointer (SP) is register R13. In Thread mode, the function of this register changes depending on the ASP bit in the Control Register (CONTROL) register. When the ASP bit is clear, this register is the Main Stack Pointer (MSP). When the ASP bit is set, this register is the Process Stack Pointer (PSP). On reset, the ASP bit is clear, and the processor loads the MSP with the value from address 0x0000.0000. The MSP can only be accessed in privileged mode; the PSP can be accessed in either privileged or unprivileged mode.

#### 1.3.4.3 Register 14: Link Register (LR, R14)

- The Link Register (LR) is register R14, and it stores the return information for subroutines, function calls, and exceptions. The Link Register can be accessed from either privileged or unprivileged mode.
- The LR receives the return address from PC when a Branch and Link (BL) or Branch and Link with Exchange (BLX) instruction is executed.
- The LR is also used for exception return. EXC\_RETURN is loaded into the LR on exception entry. See [Table 1-9](#) for the values and description.
- At all other times, R14 can be treated as a general-purpose register.

#### 1.3.4.4 Register 15: Program Counter (PC, R15)

- The Program Counter (PC) is register R15, and it contains the address of the next instruction to be executed.
- On reset, the processor loads the PC with the value of the reset vector, which is at address 0x0000.0004.
- Bit 0 of the reset vector is loaded into the THUMB bit of the EPSR at reset and must be 1.
- The PC register can be accessed in either privileged or unprivileged mode.

#### 1.3.4.5 Register 16 Program Status Register (PSR)

This register is also referred to as xPSR.

The Program Status Register (PSR) has three functions, and the register bits are assigned to the different functions:

- Application Program Status Register (APSR), bits 31:27, bits 19:16
- Execution Program Status Register (EPSR), bits 26:24, 15:10
- Interrupt Program Status Register (IPSR), bits 7:0

The PSR IPSR, and EPSR registers can only be accessed in privileged mode; the APSR register can be accessed in either privileged or unprivileged mode.

APSR contains the current state of the condition flags from previous instruction executions.

EPSR contains the Thumb state bit and the execution state bits for the If-Then (IT) instruction or the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction. Attempts to read the EPSR directly through application software using the MSR instruction always return zero. Attempts to write the EPSR using the MSR instruction in application software are always ignored. Fault handlers can examine the EPSR value in the stacked PSR to determine the operation that faulted (see [Section 1.5.8](#)).

IPSR contains the exception type number of the current Interrupt Service Routine (ISR).

These registers can be accessed individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example, all of the registers can be read using PSR with the MRS instruction, or APSR only can be written to using APSR with the MSR instruction. [Table 1-4](#) shows the possible register combinations for the PSR. See the MRS and MSR instruction descriptions in the Cortex-M4 instruction set chapter in the [Cortex-M4 Devices Generic User Guide](#) for more information about how to access the program status registers.

**Table 1-4. PSR Register Combinations**

Register	Type	Combination
PSR	RW <sup>(1)(2)</sup>	APSR, EPSR, and IPSR
IEPSR	RO	EPSR and IPSR
IAPSR	RW <sup>(1)</sup>	APSR and IPSR
EAPSR	RW <sup>(2)</sup>	APSR and EPSR

<sup>(1)</sup> The processor ignores writes to the IPSR bits.

<sup>(2)</sup> Reads of the EPSR bits return zero, and the processor ignores writes to these bits.

#### 1.3.4.6 Register 17: Priority Mask Register (PRIMASK)

The PRIMASK register prevents activation of all exceptions with programmable priority. Reset, nonmaskable interrupt (NMI), and hard fault are the only exceptions with fixed priority. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the PRIMASK register, and the CPS instruction may be used to change the value of the PRIMASK register. See the Cortex-M4 instruction set chapter in the [Cortex-M4 Devices Generic User Guide](#) for more information on these instructions. For more information on exception priority levels, see [Section 1.5.2](#).

#### 1.3.4.7 Register 18: FaultMask Register (FAULTMASK)

The FAULTMASK register prevents activation of all exceptions except for the Non-Maskable Interrupt (NMI). Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the FAULTMASK register, and the CPS instruction may be used to change the value of the FAULTMASK register. See the Cortex-M4 instruction set chapter in the [Cortex-M4 Devices Generic User Guide](#) for more information on these instructions. For more information on exception priority levels, see [Section 1.5.2](#).

#### 1.3.4.8 Register 19: Base Priority Mask Register (BASEPRI)

The BASEPRI register defines the minimum priority for exception processing. When BASEPRI is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the BASEPRI value. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. For more information on exception priority levels, see [Section 1.5.2](#).

#### 1.3.4.9 Register 20: Control Register (CONTROL)

The CONTROL register controls the stack used and the privilege level for software execution when the processor is in Thread mode, and indicates whether the FPU state is active. This register is only accessible in privileged mode.

Handler mode always uses the MSP, so the processor ignores explicit writes to the ASP bit of the CONTROL register when in Handler mode. The exception entry and return mechanisms automatically update the CONTROL register based on the EXC\_RETURN value (see [Table 1-9](#)). In an Operating system (OS), for example TI-RTOS environment, threads running in Thread mode should use the process stack and the kernel and exception handlers should use the main stack. By default, Thread mode uses the MSP. To switch the stack pointer used in Thread mode to the PSP, either use the MSR instruction to set the ASP bit, as detailed in the Cortex-M4 instruction set chapter in the [Cortex-M4 Devices Generic User Guide](#), or perform an exception return to Thread mode with the appropriate EXC\_RETURN value, as shown in [Table 1-9](#).

---

**NOTE:** When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction, ensuring that instructions after the ISB execute use the new stack pointer. See the Cortex-M4 instruction set chapter in the [Cortex-M4 Devices Generic User Guide](#).

---

#### 1.3.4.10 Register 21: Floating-Point Status Control (FPSC)

The FPSC register provides all necessary user-level control of the floating-point system.

#### 1.3.5 Exceptions and Interrupts

The Cortex-M4F processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses Handler mode to handle all exceptions except for reset. See [Section 1.5](#) for more information.

The NVIC registers control interrupt handling. See [Section 2.2.2](#) for more information.

#### 1.3.6 Data Types

The Cortex-M4F supports 32-bit words, 16-bit halfwords, and 8-bit bytes. The processor also supports 64-bit data transfer instructions. All instruction and data memory accesses are little endian. See [Section 1.4.1](#) for more information.

### 1.4 Memory Model

This section describes the processor memory map, the behavior of memory accesses, and the bit-banding features. The processor has a fixed memory map that provides up to 4GB of addressable memory. The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data (see [Section 1.4.5](#)).

The processor reserves regions of the Private Peripheral Bus (PPB) address range for core peripheral registers (see the [Cortex-M4 Peripherals](#) chapter).

---

**NOTE:** Within the memory map, attempts to read or write addresses in reserved spaces result in a bus fault.

---

---

**NOTE:** For details about the memory map of individual peripherals and valid memory range, refer to device specific datasheet.

---

#### 1.4.1 Memory Regions, Types, and Attributes

The memory map and the programming of the MPU split the memory map into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

- Normal: The processor can re-order transactions for efficiency and perform speculative reads.
- Device: The processor preserves transaction order relative to other transactions to Device or Strongly Ordered memory.
- Strongly Ordered: The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly Ordered memory mean that the memory system can buffer a write to Device memory but must not buffer a write to Strongly Ordered memory.

An additional memory attribute is Execute Never (XN), which means the processor prevents instruction accesses. A fault exception is generated only on execution of an instruction executed from an XN region.

#### 1.4.2 Memory System Ordering of Memory Accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not ensure that the order in which the accesses complete matches the program order of the instructions, providing the order does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions (see [Section 1.4.4](#)).

However, the memory system does ensure ordering of accesses to Device and Strongly Ordered memory. For two memory access instructions A1 and A2, if both A1 and A2 are accesses to either Device or Strongly Ordered memory, and if A1 occurs before A2 in program order, A1 is always observed before A2.

#### 1.4.3 Behavior of Memory Accesses

[Table 1-5](#) shows the behavior of accesses to each region in the memory map. See [Section 1.4.1](#) for more information on memory types and the XN attribute. MSP432P4xx devices may have reserved memory areas within the address ranges shown below (refer to device datasheet for more information).

**Table 1-5. Memory Access Behavior**

Address Range	Memory Region	Memory Type	Execute Never (XN)	Description
0x0000_0000–0x1FFF_FFFF	Code	Normal	–	This executable region is for program code. Data can also be stored here.
0x2000_0000–0x3FFF_FFFF	SRAM	Normal	–	This executable region is for data. This region includes bit band and bit band alias areas (see <a href="#">Table 1-6</a> ).
0x4000_0000–0x5FFF_FFFF	Peripheral	Device	XN	This region includes bit band and bit band alias areas (see <a href="#">Table 1-7</a> ).
0x6000_0000–0xDFFF_FFFF	Reserved	Normal	–	Reserved in MSP432P4xx.
0xE000_0000–0xE00F_FFFF	Private peripheral bus	Strongly Ordered	XN	This region includes the NVIC, system timer, and system control block.
0xE010_0000–0xFFFF_FFFF	Reserved	–	–	–

The MPU can override the default memory access behavior described in this section. For more information, see [Section 2.2.4](#).

The Cortex-M4F prefetches instructions ahead of execution and speculatively prefetches from branch target addresses.

#### 1.4.4 Software Ordering of Memory Accesses

The order of instructions in the program flow does not always ensure the order of the corresponding memory transactions for the following reasons:

- The processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- The processor has multiple bus interfaces.
- Memory or devices in the memory map have different wait states.

- Some memory accesses are buffered or speculative.
- [Section 1.4.2](#) describes the cases where the memory system ensures the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The Cortex-M4F has the following memory barrier instructions:
- The Data Memory Barrier (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions.
- The Data Synchronization Barrier (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute.
- The Instruction Synchronization Barrier (ISB) instruction ensures that the effect of all completed memory transactions is recognizable by subsequent instructions.

Memory barrier instructions can be used in the following situations:

- MPU programming  
If the MPU settings are changed and the change must be effective on the very next instruction, use a DSB instruction to ensure the effect of the MPU takes place immediately at the end of context switching.  
Use an ISB instruction to ensure the new MPU setting takes effect immediately after programming the MPU region or regions, if the MPU configuration code was accessed using a branch or call. If the MPU configuration code is entered using exception mechanisms, then an ISB instruction is not required.
- Vector table  
If the program changes an entry in the vector table and then enables the corresponding exception, use a DMB instruction between the operations. The DMB instruction ensures that if the exception is taken immediately after being enabled, the processor uses the new exception vector.
- Self-modifying code  
If a program contains self-modifying code, use an ISB instruction immediately after the code modification in the program. The ISB instruction ensures subsequent instruction execution uses the updated program.
- Memory map switching  
If the system contains a memory map switching mechanism, use a DSB instruction after switching the memory map in the program. The DSB instruction ensures subsequent instruction execution uses the updated memory map.
- Dynamic exception priority change  
When an exception priority has to change when the exception is pending or active, use DSB instructions after the change. The change then takes effect on completion of the DSB instruction.

Memory accesses to Strongly Ordered memory, such as the System Control Block, do not require the use of DMB instructions.

For more information on the memory barrier instructions, see the Cortex-M4 instruction set chapter in the [Cortex-M4 Devices Generic User Guide](#).

#### 1.4.5 Bit-Banding

A bit-band region maps each word in a bit-band alias region to a single bit in the bit-band region. The bit-band regions occupy the lowest 1MB of the SRAM and peripheral memory regions. Accesses to the 32MB SRAM alias region map to the 1MB SRAM bit-band region, as shown in [Table 1-6](#). Accesses to the 32MB peripheral alias region map to the 1MB peripheral bit-band region, as shown in [Table 1-7](#).

---

**NOTE:** A word access to the SRAM or the peripheral bit-band alias region maps to a single bit in the SRAM or peripheral bit-band region.

---

A word access to a bit band address results in a word access to the underlying memory, and similarly for halfword and byte accesses. This allows bit band accesses to match the access requirements of the underlying peripheral.

**Table 1-6. SRAM Memory Bit-Banding Regions**

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x2000_0000	0x2000_7FFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias.
0x2200_0000	0x220F_FFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

**Table 1-7. Peripheral Memory Bit-Banding Regions**

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x4000_0000	0x400F_FFFF	Peripheral bit-band region	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.
0x4200_0000	0x43FF_FFFF	Peripheral bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

The following formula shows how the alias region maps onto the bit-band region:

$$\begin{aligned} \text{bit\_word\_offset} &= (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4) \\ \text{bit\_word\_addr} &= \text{bit\_band\_base} + \text{bit\_word\_offset} \end{aligned}$$

where:

**bit\_word\_offset**

The position of the target bit in the bit-band memory region.

**bit\_word\_addr**

The address of the word in the alias memory region that maps to the targeted bit.

**bit\_band\_base**

The starting address of the alias region.

**byte\_offset**

The number of the byte in the bit-band region that contains the targeted bit.

**bit\_number**

The bit position, 0-7, of the targeted bit.

Figure 1-3 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

- The alias word at 0x23FF\_FFE0 maps to bit 0 of the bit-band byte at 0x200F\_FFFF:

$$0x23FF.FFE0 = 0x2200_0000 + (0x000F.FFFF * 32) + (0 * 4)$$

- The alias word at 0x23FF\_FFFC maps to bit 7 of the bit-band byte at 0x200F\_FFFF:

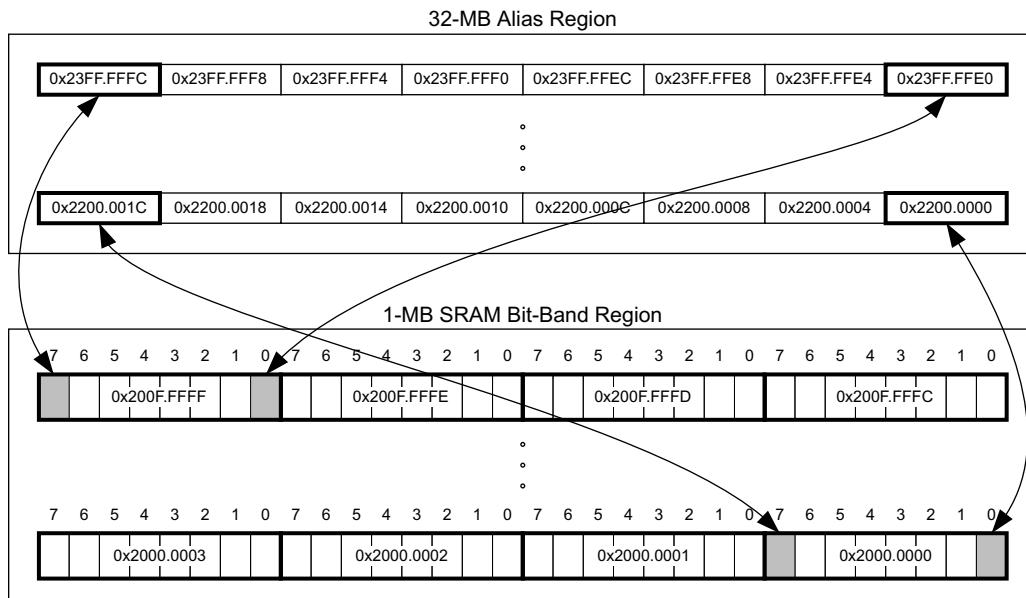
$$0x23FF.FFFC = 0x2200_0000 + (0x000F.FFFF * 32) + (7 * 4)$$

- The alias word at 0x2200\_0000 maps to bit 0 of the bit-band byte at 0x2000\_0000:

$$0x2200.0000 = 0x2200_0000 + (0 * 32) + (0 * 4)$$

- The alias word at 0x2200\_001C maps to bit 7 of the bit-band byte at 0x2000\_0000:

$$0x2200.001C = 0x2200_0000 + (0 * 32) + (7 * 4)$$



**Figure 1-3. Bit-Band Mapping**

#### 1.4.5.1 Directly Accessing an Alias Region

Writing to a word in the alias region updates a single bit in the bit-band region.

Bit 0 of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit 0 set writes a 1 to the bit-band bit, and writing a value with bit 0 clear writes a 0 to the bit-band bit.

Bits 31:1 of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0xE.

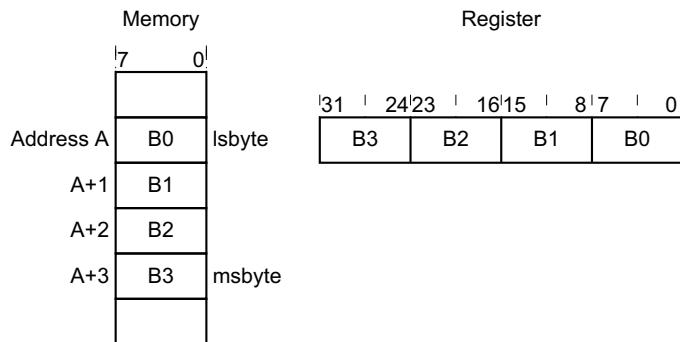
When reading a word in the alias region, 0x0000\_0000 indicates that the targeted bit in the bit-band region is clear and 0x0000\_0001 indicates that the targeted bit in the bit-band region is set.

#### 1.4.5.2 Directly Accessing a Bit-Band Region

[Section 1.4.3](#) describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

#### 1.4.6 Data Storage

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Data is stored in little-endian format, with the least-significant byte (LSByte) of a word stored at the lowest-numbered byte, and the most-significant byte (MSByte) stored at the highest-numbered byte. [Figure 1-4](#) shows how data is stored.



**Figure 1-4. Data Storage**

## 1.5 Exception Model

The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. When handling exceptions:

- All exceptions are handled in Handler mode.
- Processor state is automatically stored to the stack on an exception, and automatically restored from the stack at the end of the Interrupt Service Routine (ISR).
- The vector is fetched in parallel to the state saving, enabling efficient interrupt entry.
- The processor supports tail-chaining that enables back-to-back interrupts without the overhead of state saving and restoration.

Software can choose only to enable a subset of the configured number of interrupts, and can choose how many bits of the configured priorities to use.

Internally, the highest user-programmable priority (0) is treated as fourth priority, after a Reset, Non-Maskable Interrupt (NMI), and a Hard Fault, in that order. 0 is the default priority for all the programmable priorities.

---

**NOTE:** Vector table entries are compatible with interworking between ARM and Thumb instructions. This causes bit [0] of the vector value to load into the Execution Program Status register (EPSR) T-bit on exception entry. All populated vectors in the vector table entries must have bit [0] set. Creating a table entry with bit [0] clear generates an INVSTATE fault on the first instruction of the handler corresponding to this vector.

---



---

**NOTE:** After a write to clear an interrupt source, it may take several processor cycles for the NVIC to see the interrupt source deassert. Thus if the interrupt clear is done as the last action in an interrupt handler, it is possible for the interrupt handler to complete while the NVIC sees the interrupt as still asserted, causing the interrupt handler to be re-entered errantly. This situation can be avoided by either clearing the interrupt source at the beginning of the interrupt handler or by performing a read or write after the write to clear the interrupt source (and flush the write buffer).

---

See [Section 2.2.2](#) for more information on exceptions and interrupts.

### 1.5.1 Exception States

Each exception is in one of the following states:

- Inactive: The exception is not active and not pending.
- Pending: The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
- Active: An exception that is being serviced by the processor but has not completed.

---

**NOTE:** An exception handler can interrupt the execution of another exception handler. In this case, both exceptions are in the active state.

- Active and Pending: The exception is being serviced by the processor, and there is a pending exception from the same source.

### 1.5.2 Exception Types

The exception types are:

- Reset: Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in Thread mode.
- NMI: A nonmaskable Interrupt (NMI) can be signaled using the NMI signal or triggered by software using the Interrupt Control State register (ICSR). This exception has the highest priority other than reset. NMI is permanently enabled and has a fixed priority of -2. NMIs cannot be masked or prevented from activation by any other exception or preempted by any exception other than reset.
- Hard Fault: A hard fault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. Hard faults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.
- Memory Management Fault: A memory management fault is an exception that occurs because of a memory protection related fault, including access violation and no match. The MPU or the fixed memory protection constraints determine this fault, for both instruction and data memory transactions. This fault is used to abort instruction accesses to Execute Never (XN) memory regions, even if the MPU is disabled.
- Bus Fault: A bus fault can occur under various conditions as listed below
  - Accessing a vacant address in the memory map
  - Accessing 16-bit peripherals with 32-bit accesses
  - Any other condition now allowed by the system e.g. write access to ROM memory
- Usage Fault: A usage fault is an exception that occurs because of a fault related to instruction execution, such as:
  - An undefined instruction
  - An illegal unaligned access
  - Invalid state on instruction execution
  - An error on exception return

An unaligned address on a word or halfword memory access or division by zero can cause a usage fault when the core is properly configured.

- SVCALL: A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.
- Debug Monitor: This exception is caused by the debug monitor (when not halting). This exception is only active when enabled. This exception does not activate if it is a lower priority than the current activation.
- PendSV: PendSV is a pendable, interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active. PendSV is triggered using the Interrupt Control State register (ICSR).
- SysTick: A SysTick exception is an exception that the system timer generates when it reaches zero

when it is enabled to generate an interrupt. Software can also generate a SysTick exception using the Interrupt Control State register (ICSR). In an OS environment, the processor can use this exception as system tick.

- **Interrupt (IRQ):** An interrupt, or IRQ, is an exception signaled by a peripheral or generated by a software request and fed through the NVIC (prioritized). All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor.

For an asynchronous exception, other than reset, the processor can execute another instruction between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that [Table 1-8](#) shows as having configurable priority (see the SHCSR register and the ICER0 register).

For more information about hard faults, memory management faults, bus faults, and usage faults, see [Section 1.6](#).

**Table 1-8. Exception Types**

Exception Type	Vector Number	Priority <sup>(1)</sup>	Vector Address or Offset <sup>(2)</sup>	Activation
-	0	-	0x0000_0000	Stack top is loaded from the first entry of the vector table on reset.
Reset	1	-3 (highest)	0x0000_0004	Asynchronous
Non-Maskable Interrupt (NMI)	2	-2	0x0000.0008	Asynchronous
Hard Fault	3	-1	0x0000_000C	-
Memory Management	4	Programmable <sup>(3)</sup>	0x0000_0010	Synchronous
Bus Fault	5	Programmable <sup>(3)</sup>	0x0000_0014	Synchronous when precise and asynchronous when imprecise
Usage Fault	6	Programmable <sup>(3)</sup>	0x0000_0018	Synchronous
-	7-10	-	-	Reserved
SVCall	11	Programmable <sup>(3)</sup>	0x0000_002C	Synchronous
Debug Monitor	12	Programmable <sup>(3)</sup>	0x0000_0030	Synchronous
-	13	-	-	Reserved
PendSV	14	Programmable <sup>(3)</sup>	0x0000_0038	Asynchronous
SysTick	15	Programmable <sup>(3)</sup>	0x0000_003C	Asynchronous
Interrupts	16 and above	Programmable <sup>(4)</sup>	0x0000_0040 and above	Asynchronous

<sup>(1)</sup> 0 is the default priority for all the programmable priorities.

<sup>(2)</sup> See [Section 1.5.4](#).

<sup>(3)</sup> See SHPR1 register.

<sup>(4)</sup> See IPRn registers.

---

**NOTE:** For details about the mapping of peripheral interrupts, refer to appropriate device datasheet.

---

### 1.5.3 Exception Handlers

The processor handles exceptions using:

- **Interrupt Service Routines (ISRs):** Interrupts (IRQx) are the exceptions handled by ISRs.
- **Fault Handlers:** Hard fault, memory management fault, usage fault, and bus fault are fault exceptions handled by the fault handlers.
- **System Handlers:** NMI, PendSV, SVCall, SysTick, and the fault exceptions are all system exceptions that are handled by system handlers.

### 1.5.4 Vector Table

The vector table contains the reset value of the stack pointer and the start addresses, also called exception vectors, for all exception handlers. The vector table is constructed using the vector address or offset shown in [Table 1-8](#). [Figure 1-5](#) shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code.

Exception number	IRQ number	Offset	Vector
154	138	0x0268	IRQ131
.	.	.	.
.	.	.	.
.	.	.	.
18	2	0x004C	IRQ2
17	1	0x0048	IRQ1
16	0	0x0044	IRQ0
15	-1	0x0040	Systick
14	-2	0x003C	PendSV
13		0x0038	Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			
9			Reserved
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

**Figure 1-5. Vector Table**

On system reset, the vector table is fixed at address 0x0000.0000. Privileged software can write to the Vector Table Offset Register (VTOR) to relocate the vector table start address to a different memory location, in the range 0x0000\_0000 to 0x3FFF\_FFFF. When configuring the Vector Table Offset Register (VTOR), the offset must be aligned on a 512-byte boundary.

### 1.5.5 Exception Priorities

As [Table 1-8](#) shows, all exceptions have an associated priority. A smaller number for the priority value indicates a higher priority level. All exceptions have configurable priorities except Reset, Hard fault, and NMI. If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities, see the SHPR1 and IPRn registers.

---

**NOTE:** Configurable priority values for the MSP432P4xx series implementation are in the range of 0 to 7.

---

This means that the Reset, Hard fault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, assigning a priority value of 5 to IRQ[0] and a priority value of 4 to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

### 1.5.6 Interrupt Priority Grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This grouping divides each interrupt priority register entry into two fields:

- An upper field that defines the group priority
- A lower field that defines a subpriority within the group

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler.

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

For information about splitting the interrupt priority fields into group priority and subpriority, see the Application Interrupt and Reset Control (AIRCR) register.

### 1.5.7 Level and Pulse Interrupts

The processor supports both level and pulse interrupts. A level interrupt is held asserted until it is cleared by the ISR accessing the device. A pulse interrupt is a variant of an edge model.

For level interrupts, if the signal is not deasserted before the return from the interrupt routine, the interrupt again enters the pending state and re-activates. This is particularly useful for FIFO and buffer-based devices because it ensures that they drain either by a single ISR or by repeated invocations, with no extra work. This means that the device holds the signal in assert until the device is empty.

A pulse interrupt can be reasserted during the ISR so that the interrupt can be in the pending state and active at the same time. If another pulse arrives while the interrupt is still pending, the interrupt remains pending and the ISR runs only once.

Pulse interrupts are mostly used for external signals and for rate or repeat signals.

### 1.5.8 Exception Entry and Return

The processor implements advanced exception and interrupt handling, as described in the [ARMv7-M Architecture Reference Manual \(ARM DDI0403E.B\)](#).

To reduce interrupt latency, the processor implements both interrupt late-arrival and interrupt tail-chaining mechanisms:

- There is a maximum of twelve cycle latency from asserting the interrupt to execution of the first instruction of the ISR when the memory being accessed has no wait states being applied. The first instruction to be executed is fetched in parallel to the stack push.
- Returns from interrupts takes twelve cycles where the instruction being returned to is fetched in parallel to the stack pop.
- Tail chaining requires six cycles when using zero wait state memory. No stack pushes or pops are performed and only the instruction for the next ISR is fetched.

To minimize interrupt latency, the processor abandons any divide instruction to take any pending interrupt. On return from the interrupt handler, the processor restarts the divide instruction from the beginning. The processor implements the Interruptible-Continuable Instruction field. Load multiple (LDM) operations and store multiple (STM) operations are interruptible. The EPSR holds the information required to continue the load or store multiple from the point where the interrupt occurred.

Descriptions of exception handling use the following terms:

- Preemption: When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See [Section 1.5.6](#) for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See [Section 1.5.8.1](#) for more information.
- Return: Return occurs when the exception handler is completed, and there is no pending exception with sufficient priority to be serviced and the completed exception handler was not handling a late-arriving exception. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See [Section 1.5.8.2](#) for more information.
- Tail-Chaining: This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.
- Late-Arriving: This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore, the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

#### 1.5.8.1 Exception Entry

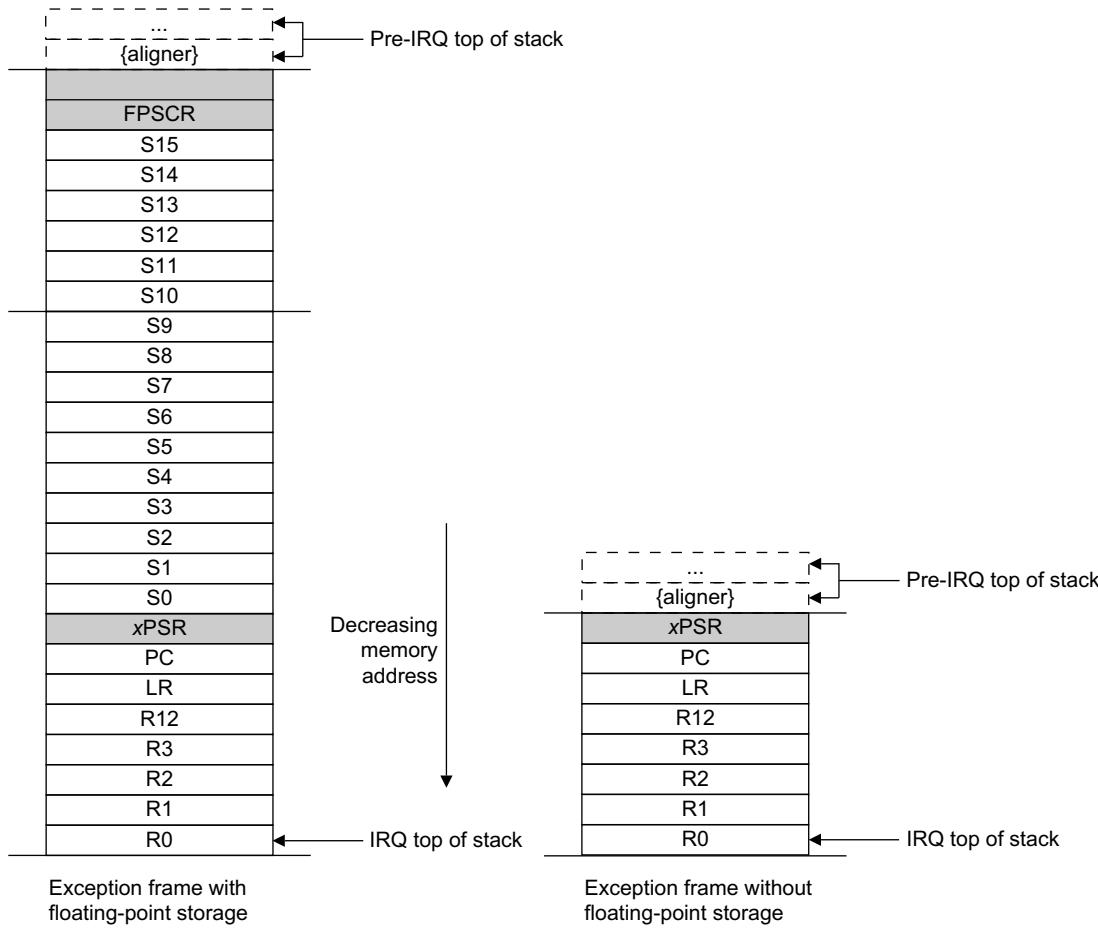
Exception entry occurs when there is a pending exception with sufficient priority and either the processor is in Thread mode or the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception. When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has more priority than any limits set by the mask registers (see the PRIMASK, FAULTMASK, and BASEPRI registers). An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as *stacking* and the structure of eight data words is referred to as *stack frame*.

When using floating-point routines, the Cortex-M4F processor automatically stacks the architected floating-point state on exception entry. [Figure 1-6](#) shows the Cortex-M4F stack frame layout when floating-point state is preserved on the stack as the result of an interrupt or an exception.

**NOTE:** Where stack space for floating-point state is not allocated, the stack frame is the same as that of ARMv7-M implementations without an FPU. [Figure 1-6](#) shows this stack frame also.



**Figure 1-6. Exception Stack Frame**

Immediately after stacking, the stack pointer indicates the lowest address in the stack frame.

The stack frame includes the return address, which is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes.

In parallel with the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC\_RETURN value to the LR, indicating which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher-priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher-priority exception occurs during exception entry, known as late arrival, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception.

### 1.5.8.2 Exception Return

Exception return occurs when the processor is in Handler mode and executes one of the following instructions to load the EXC\_RETURN value into the PC:

- An LDM or POP instruction that loads the PC
- A BX instruction using any register
- An LDR instruction with the PC as the destination

EXC\_RETURN is the value loaded into the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest five bits of this value provide information on the return stack and processor mode. [Table 1-9](#) shows the EXC\_RETURN values with a description of the exception return behavior.

EXC\_RETURN bits 31:5 are all set. When this value is loaded into the PC, it indicates to the processor that the exception is complete, and the processor initiates the appropriate exception return sequence.

**Table 1-9. Exception Return Behavior**

EXC_RETURN[31:0]	Description
0xFFFF_FFE0	Reserved
0xFFFF_FFE1	Return to Handler mode. Exception return uses floating-point state from MSP. Execution uses MSP after return.
0xFFFF_FFE2 - 0xFFFF_FFE8	Reserved
0xFFFF_FFE9	Return to Thread mode. Exception return uses floating-point state from MSP. Execution uses MSP after return.
0xFFFF_FFEA - 0xFFFF_FFEC	Reserved
0xFFFF_FFED	Return to Thread mode. Exception return uses floating-point state from PSP. Execution uses PSP after return.
0xFFFF_FFEE - 0xFFFF_FFF0	Reserved
0xFFFF_FFF1	Return to Handler mode. Exception return uses non-floating-point state from MSP. Execution uses MSP after return.
0xFFFF_FFF2 - 0xFFFF_FFF8	Reserved
0xFFFF_FFF9	Return to Thread mode. Exception return uses non-floating-point state from MSP. Execution uses MSP after return.
0xFFFF_FFFA - 0xFFFF_FFFC	Reserved
0xFFFF_FFFD	Return to Thread mode. Exception return uses non-floating-point state from PSP. Execution uses PSP after return.
0xFFFF_FFFE - 0xFFFF_FFFF	Reserved

## 1.6 Fault Handling

Faults are a subset of the exceptions (see [Section 1.5](#)). The following conditions generate a fault:

- A bus error on an instruction fetch or vector table load or a data access.
- An internally detected error such as an undefined instruction or an attempt to change state with a BX instruction.
- Attempting to execute an instruction from a memory region marked as Non-Executable (XN).
- An MPU fault because of a privilege violation or an attempt to access an unmanaged region.

### 1.6.1 Fault Types

**Table 1-10** shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates the fault has occurred. See the FAULTSTAT register for more information.

**Table 1-10. Faults**

Fault	Handler	Fault Status Register	Bit Name
Bus error on a vector read	Hard fault	Hard Fault Status (HFSR)	VECTTBL
Fault escalated to a hard fault	Hard fault	Hard Fault Status (HFSR)	FORCED
MPU or default memory mismatch on instruction access	Memory management fault	Memory Manage Fault Status Register (MMFSR)	IACCVIOL <sup>(1)</sup>
MPU or default memory mismatch on data access	Memory management fault	Memory Manage Fault Status Register (MMFSR)	DACCVIOL
MPU or default memory mismatch on exception stacking	Memory management fault	Memory Manage Fault Status Register (MMFSR)	MSTKERR
MPU or default memory mismatch on exception unstacking	Memory management fault	Memory Manage Fault Status Register (MMFSR)	MUNSTKERR
MPU or default memory mismatch during lazy floating-point state preservation	Memory management fault	Memory Manage Fault Status Register (MMFSR)	MLSPERR
Bus error during exception stacking	Bus fault	Bus Fault Status Register (BFSR)	STKERR
Bus error during exception unstacking	Bus fault	Bus Fault Status Register (BFSR)	UNSTKERR
Bus error during instruction prefetch	Bus fault	Bus Fault Status Register (BFSR)	IBUSERR
Bus error during lazy floating-point state preservation	Bus fault	Bus Fault Status Register (BFSR)	BLSPERR
Precise data bus error	Bus fault	Bus Fault Status Register (BFSR)	PRECISERR
Imprecise data bus error	Bus fault	Bus Fault Status Register (BFSR)	IMPRECISERR
Attempt to access a coprocessor	Usage fault	Usage Fault Status Register (UFSR)	NOCP
Undefined instruction	Usage fault	Usage Fault Status Register (UFSR)	UNDEFINSTR
Attempt to enter an invalid instruction set state <sup>(2)</sup>	Usage fault	Usage Fault Status Register (UFSR)	INVSTATE
Invalid EXC_RETURN value	Usage fault	Usage Fault Status Register (UFSR)	INVPC
Illegal unaligned load or store	Usage fault	Usage Fault Status Register (UFSR)	UNALIGNED
Divide by 0	Usage fault	Usage Fault Status Register (UFSR)	DIVBYZERO

<sup>(1)</sup> Occurs on an access to an XN region even if the MPU is disabled.

<sup>(2)</sup> Attempting to use an instruction set other than the Thumb instruction set, or returning to a non load-store-multiply instruction with ICI continuation.

### 1.6.2 Fault Escalation and Hard Faults

All fault exceptions except for hard fault have configurable exception priority (see the SHPR1 register). Software can disable execution of the handlers for these faults (see the SHCSR register).

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler (see [Section 1.5](#)).

In some situations, a fault with configurable priority is treated as a hard fault. This process is called priority escalation, and the fault is described as *escalated to hard fault*. Escalation to hard fault occurs when:

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to hard fault occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level.
- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This situation happens because the handler for the new fault cannot preempt the currently executing fault handler.
- An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception.
- A fault occurs and the handler for that fault is not enabled.

If a bus fault occurs during a stack push when entering a bus fault handler, the bus fault does not escalate to a hard fault. Thus if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted.

---

**NOTE:** Only Reset and NMI can preempt the fixed priority hard fault. A hard fault can preempt any exception other than Reset, NMI, or another hard fault.

---

### 1.6.3 Fault Status Registers and Fault Address Registers

The fault status registers indicate the cause of a fault. For bus faults and memory management faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in [Table 1-11](#).

**Table 1-11. Fault Status and Fault Address Registers**

Handler	Status Register Name	Address Register Name
Hard fault	Hard Fault Status Register (HFSR)	Hard Fault Status Register (HFSR)
Memory management fault	Memory Management Fault Status Register (MMFSR)	Configurable Fault Status Register (CFSR)
Bus fault	Bus Fault Status Register (BFSR)	Configurable Fault Status Register (CFSR)
Usage fault	Usage Fault Status Register (UFSR)	Configurable Fault Status Register (CFSR)

---

**NOTE:** MMFSR, BFSR and UFSR together form the Configurable Fault Status Register (CFSR). Refer to the register section for more details.

---

### 1.6.4 Lockup

The processor enters a lockup state if a hard fault occurs when executing the NMI or hard fault handlers. When the processor is in the lockup state, it does not execute any instructions. The processor remains in lockup state until it is reset, an NMI occurs, or it is halted by a debugger.

---

**NOTE:** If the lockup state occurs from the NMI handler, a subsequent NMI does not cause the processor to leave the lockup state.

---

## 1.7 Power Management

The Cortex-M4F processor sleep modes reduce power consumption:

- Sleep mode stops the processor clock.
- Deep-sleep mode switches off the Flash memory, powers down the digital logic and puts under retention.

The processor fully implements the Wait For Interrupt (WFI) and Wait For Event (WFE). In addition, the processor also supports the use of SLEEPONEXIT, that causes the processor core to enter sleep mode when it returns from an exception handler to Thread mode.

MSP432P4xx implementation also supports the processor to be put into various very low-power sleep modes and capability to wakeup from certain predefined events.

---

**NOTE:** For details on the mechanisms of entry to and exit from various low-power modes, refer to the Power Control Manager (PCM) chapter.

---

## 1.8 Instruction Set Summary

The processor implements a version of the Thumb instruction set. [Table 1-12](#) lists the supported instructions.

**NOTE:** In [Table 1-12](#):

- Angle brackets, <>, enclose alternative forms of the operand
- Braces, {}, enclose optional operands
- The Operands column is not exhaustive
- Op2 is a flexible second operand that can be either a register or a constant
- Most instructions can use an optional condition code suffix

For more information on the instructions and operands, see the instruction descriptions in the *ARM® Cortex-M4 Technical Reference Manual*.

**Table 1-12. Cortex-M4F Instruction Summary**

Mnemonic	Operands	Brief Description	Flags
ADC, ADCS	{Rd,} Rn, Op2	Add with Carry	N, Z, C, V
ADD, ADDS	{Rd,} Rn, Op2	Add	N, Z, C, V
ADD, ADDW	{Rd,} Rn, #imm12	Add	—
ADR	Rd, label	Load PC-relative Address	—
AND, ANDS	{Rd,} Rn, Op2	Logical AND	N, Z, C
ASR, ASRS	Rd, Rm, <Rs #n>	Arithmetic Shift Right	N, Z, C
B	label	Branch	—
BFC	Rd, #lsb, #width	Bit Field Clear	—
BFI	Rd, Rn, #lsb, #width	Bit Field Insert	—
BIC, BICS	{Rd,} Rn, Op2	Bit Clear	N, Z, C
BKPT	#imm	Breakpoint	—
BL	label	Branch with Link	—
BLX	Rm	Branch indirect with Link	—
BX	Rm	Branch indirect	—
CBNZ	Rn, label	Compare and Branch if Non Zero	—
CBZ	Rn, label	Compare and Branch if Zero	—
CLREX	—	Clear Exclusive	—
CLZ	Rd, Rm	Count Leading Zeros	—
CMN	Rn, Op2	Compare Negative	N, Z, C, V
CMP	Rn, Op2	Compare	N, Z, C, V
CPSID	i	Change Processor State, Disable Interrupts	—
CPSIE	i	Change Processor State, Enable Interrupts	—
DMB	—	Data Memory Barrier	—
DSB	—	Data Synchronization Barrier	—
EOR, EORS	{Rd,} Rn, Op2	Exclusive OR	N, Z, C
ISB	—	Instruction Synchronization Barrier	—
IT	—	If-Then condition block	—
LDM	Rn{!}, reglist	Load Multiple registers, increment after	—
LDMDB, LDMEA	Rn{!}, reglist	Load Multiple registers, decrement before	—
LDMFD, LDMIA	Rn{!}, reglist	Load Multiple registers, increment after	—
LDR	Rt, [Rn, #offset]	Load Register with word	—
LDRB, LDRBT	Rt, [Rn, #offset]	Load Register with byte	—
LDRD	Rt, Rt2, [Rn, #offset]	Load Register with two bytes	—
LDREX	Rt, [Rn, #offset]	Load Register Exclusive	—

**Table 1-12. Cortex-M4F Instruction Summary (continued)**

Mnemonic	Operands	Brief Description	Flags
LDREXB	Rt, [Rn]	Load Register Exclusive with Byte	–
LDREXH	Rt, [Rn]	Load Register Exclusive with Halfword	–
LDRH, LDRHT	Rt, [Rn, #offset]	Load Register with Halfword	–
LDRSB, LDRSBT	Rt, [Rn, #offset]	Load Register with Signed Byte	–
LDRSH, LDRSHT	Rt, [Rn, #offset]	Load Register with Signed Halfword	–
LDRT	Rt, [Rn, #offset]	Load Register with word	–
LSL, LSLS	Rd, Rm, <Rs #n>	Logical Shift Left	N, Z, C
LSR, LSRS	Rd, Rm, <Rs #n>	Logical Shift Right	N, Z, C
MLA	Rd, Rn, Rm, Ra	Multiply with Accumulate, 32-bit result	–
MLS	Rd, Rn, Rm, Ra	Multiply and Subtract, 32-bit result	–
MOV, MOVS	Rd, Op2	Move	N, Z, C
MOVT	Rd, #imm16	Move Top	–
MOVW, MOV	Rd, #imm16	Move 16-bit constant	N, Z, C
MRS	Rd, spec_reg	Move from Special Register to general register	–
MSR	spec_reg, Rm	Move from general register to Special Register	N, Z, C, V
MUL, MULS	{Rd,} Rn, Rm	Multiply, 32-bit result	N,Z
MVN, MVNS	Rd, Op2	Move NOT	N, Z, C
NOP	–	No Operation	–
ORN, ORNS	{Rd,} Rn, Op2	Logical OR NOT	N, Z, C
ORR, ORRS	{Rd,} Rn, Op2	Logical OR	N, Z, C
PKHTB, PKHBT	{Rd,} Rn, Rm, Op2	Pack Halfword	–
POP	reglist	Pop registers from stack	–
PUSH	reglist	Push registers onto stack	–
QADD	{Rd,} Rn, Rm	Saturating double and Add	Q
QADD16	{Rd,} Rn, Rm	Saturating Add 16	–
QADD8	{Rd,} Rn, Rm	Saturating Add 8	–
QASX	{Rd,} Rn, Rm	Saturating Add and Subtract with Exchange	–
QDADD	{Rd,} Rn, Rm	Saturating Add	Q
QDSUB	{Rd,} Rn, Rm	Saturating double and Subtract	Q
QSAX	{Rd,} Rn, Rm	Saturating Subtract and Add with Exchange	–
QSUB	{Rd,} Rn, Rm	Saturating Subtract	Q
QSUB16	{Rd,} Rn, Rm	Saturating Subtract 16	–
QSUB8	{Rd,} Rn, Rm	Saturating Subtract 8	–
RBIT	Rd, Rn	Reverse Bits	–
REV	Rd, Rn	Reverse byte order in a word	–
REV16	Rd, Rn	Reverse byte order in each halfword	–
REVSH	Rd, Rn	Reverse byte order in bottom halfword and sign extend	–
ROR, RORS	Rd, Rm, <Rs #n>	Rotate Right	N, Z, C
RRX, RRXS	Rd, Rm	Rotate Right with Extend	N, Z, C
RSB, RSBS	{Rd,} Rn, Op2	Reverse Subtract	N, Z, C, V
SADD16	{Rd,} Rn, Rm	Signed Add 16	GE
SADD8	{Rd,} Rn, Rm	Signed Add 8	GE
SASX	{Rd,} Rn, Rm	Signed Add and Subtract with Exchange	GE
SBC, SBCS	{Rd,} Rn, Op2	Subtract with Carry	N, Z, C, V
SBFX	Rd, Rn, #lsb, #width	Signed Bit Field Extract	–
SDIV	{Rd,} Rn, Rm	Signed Divide	–
SEL	{Rd,} Rn, Rm	Select bytes	–

**Table 1-12. Cortex-M4F Instruction Summary (continued)**

Mnemonic	Operands	Brief Description	Flags
SEV	-	Send Event	-
SHADD16	{Rd,} Rn, Rm	Signed Halving Add 16	-
SHADD8	{Rd,} Rn, Rm	Signed Halving Add 8	-
SHASX	{Rd,} Rn, Rm	Signed Halving Add and Subtract with Exchange	-
SHSAX	{Rd,} Rn, Rm	Signed Halving Subtract and Add with Exchange	-
SHSUB16	{Rd,} Rn, Rm	Signed Halving Subtract 16	-
SHSUB8	{Rd,} Rn, Rm	Signed Halving Subtract 8	-
SMLABB, SMLABT, SMLATB, SMLATT	Rd, Rn, Rm, Ra	Signed Multiply Accumulate Long (halfwords)	Q
SMLAD, SMLADX	Rd, Rn, Rm, Ra	Signed Multiply Accumulate Dual	Q
SMLAL	RdLo, RdHi, Rn, Rm	Signed Multiply with Accumulate (32 x 32 + 64), 64-bit result	-
SMLALBB, SMLALBT, SMLALTB, SMLALTT	RdLo, RdHi, Rn, Rm	Signed Multiply Accumulate Long, halfwords	-
SMLALD, SMLALDX	RdLo, RdHi, Rn, Rm	Signed Multiply Accumulate Long Dual	-
SMLAWB, SMLAWT	Rd, Rn, Rm, Ra	Signed Multiply Accumulate, word by halfword	Q
SMLSD	Rd, Rn, Rm, Ra	Signed Multiply Subtract Dual	Q
SMLS LD	RdLo, RdHi, Rn, Rm	Signed Multiply Subtract Long Dual	
SMMLA	Rd, Rn, Rm, Ra	Signed Most significant word Multiply Accumulate	-
SMMLS, SMMLR	Rd, Rn, Rm, Ra	Signed Most significant word Multiply Subtract	-
SMMUL, SMMULR	{Rd,} Rn, Rm	Signed Most significant word Multiply	-
SMUAD	{Rd,} Rn, Rm	Signed dual Multiply Add	Q
SMULBB, SMULBT SMULTB, SMULTT	{Rd,} Rn, Rm	Signed Multiply (halfwords)	-
SMULL	RdLo, RdHi, Rn, Rm	Signed Multiply (32 x 32), 64-bit result	-
SMULWB, SMULWT	{Rd,} Rn, Rm	Signed Multiply word by halfword	-
SMUSD, SMUSDX	{Rd,} Rn, Rm	Signed dual Multiply Subtract	-
SSAT	Rd, #n, Rm {shift #s}	Signed Saturate	Q
SSAT16	Rd, #n, Rm	Signed Saturate 16	Q
SSAX	{Rd,} Rn, Rm	Signed Subtract and Add with Exchange	GE
SSUB16	{Rd,} Rn, Rm	Signed Subtract 16	-
SSUB8	{Rd,} Rn, Rm	Signed Subtract 8	-
STM	Rn{!}, reglist	Store Multiple registers, increment after	-
STMDB, STMEA	Rn{!}, reglist	Store Multiple registers, decrement before	-
STMFD, STMIA	Rn{!}, reglist	Store Multiple registers, increment after	-
STR	Rt, [Rn, #offset]	Store Register word	-
STRB, STRBT	Rt, [Rn, #offset]	Store Register byte	-
STRD	Rt, Rt2, [Rn, #offset]	Store Register two words	-
STREX	Rd, Rt, [Rn, #offset]	Store Register Exclusive	-
STREXB	Rd, Rt, [Rn]	Store Register Exclusive Byte	-
STREXH	Rd, Rt, [Rn]	Store Register Exclusive Halfword	-
STRH, STRHT	Rt, [Rn, #offset]	Store Register Halfword	-
STRT	Rt, [Rn, #offset]	Store Register word	-
SUB, SUBS	{Rd,} Rn, Op2	Subtract	N, Z, C, V
SUB, SUBW	{Rd,} Rn, #imm12	Subtract	-
SVC	#imm	Supervisor Call	-
SXTAB	{Rd,} Rn, Rm,{,ROR #}	Extend 8 bits to 32 and add	-
SXTAB16	{Rd,} Rn, Rm,{,ROR #}	Dual extend 8 bits to 16 and add	-
SXTAH	{Rd,} Rn, Rm,{,ROR #}	Extend 16 bits to 32 and add	-

**Table 1-12. Cortex-M4F Instruction Summary (continued)**

Mnemonic	Operands	Brief Description	Flags
SXTB16	{Rd,} Rm {,ROR #n}	Signed Extend Byte 16	–
SXTB	{Rd,} Rm {,ROR #n}	Sign extend a byte	–
SXTH	{Rd,} Rm {,ROR #n}	Sign extend a halfword	–
TBB	[Rn, Rm]	Table Branch Byte	–
TBH	[Rn, Rm, LSL #1]	Table Branch Halfword	–
TEQ	Rn, Op2	Test Equivalence	N, Z, C
TST	Rn, Op2	Test	N, Z, C
UADD16	{Rd,} Rn, Rm	Unsigned Add 16	GE
UADD8	{Rd,} Rn, Rm	Unsigned Add 8	GE
USAX	{Rd,} Rn, Rm	Unsigned Subtract and Add with Exchange	GE
UHADD16	{Rd,} Rn, Rm	Unsigned Halving Add 16	–
UHADD8	{Rd,} Rn, Rm	Unsigned Halving Add 8	–
UHASX	{Rd,} Rn, Rm	Unsigned Halving Add and Subtract with Exchange	–
UHSAX	{Rd,} Rn, Rm	Unsigned Halving Subtract and Add with Exchange	–
UHSUB16	{Rd,} Rn, Rm	Unsigned Halving Subtract 16	–
UHSUB8	{Rd,} Rn, Rm	Unsigned Halving Subtract 8	–
UBFX	Rd, Rn, #lsb, #width	Unsigned Bit Field Extract	–
UDIV	{Rd,} Rn, Rm	Unsigned Divide	–
UMAAL	RdLo, RdHi, Rn, Rm	Unsigned Multiply Accumulate Accumulate Long (32 x 32 + 32 + 32), 64-bit result	–
UMLAL	RdLo, RdHi, Rn, Rm	Unsigned Multiply with Accumulate (32 x 32 + 64), 64-bit result	–
UMULL	RdLo, RdHi, Rn, Rm	Unsigned Multiply (32 x 32), 64-bit result	–
UQADD16	{Rd,} Rn, Rm	Unsigned Saturating Add 16	–
UQADD8	{Rd,} Rn, Rm	Unsigned Saturating Add 8	–
UQASX	{Rd,} Rn, Rm	Unsigned Saturating Add and Subtract with Exchange	–
UQSAX	{Rd,} Rn, Rm	Unsigned Saturating Subtract and Add with Exchange	–
UQSUB16	{Rd,} Rn, Rm	Unsigned Saturating Subtract 16	–
UQSUB8	{Rd,} Rn, Rm	Unsigned Saturating Subtract 8	–
USAD8	{Rd,} Rn, Rm	Unsigned Sum of Absolute Differences	–
USADA8	{Rd,} Rn, Rm, Ra	Unsigned Sum of Absolute Differences and Accumulate	–
USAT	Rd, #n, Rm {,shift #s}	Unsigned Saturate	Q
USAT16	Rd, #n, Rm	Unsigned Saturate 16	Q
UASX	{Rd,} Rn, Rm	Unsigned Add and Subtract with Exchange	GE
USUB16	{Rd,} Rn, Rm	Unsigned Subtract 16	GE
USUB8	{Rd,} Rn, Rm	Unsigned Subtract 8	GE
UXTAB	{Rd,} Rn, Rm,{,ROR #}	Rotate, extend 8 bits to 32 and Add	–
UXTAB16	{Rd,} Rn, Rm,{,ROR #}	Rotate, dual extend 8 bits to 16 and Add	–
UXTAH	{Rd,} Rn, Rm,{,ROR #}	Rotate, unsigned extend and Add Halfword	–
UXTB	{Rd,} Rm {,ROR #n}	Zero extend a Byte	–
UXTB16	{Rd,} Rm {,ROR #n}	Unsigned Extend Byte 16	–
UXTH	{Rd,} Rm {,ROR #n}	Zero extend a Halfword	–
VABS.F32	Sd, Sm	Floating-point Absolute	–
VADD.F32	{Sd,} Sn, Sm	Floating-point Add	–
VCMP.F32	Sd, <Sm   #0.0>	Compare two floating-point registers, or one floating-point register and zero	FPSCR
VCMPE.F32	Sd, <Sm   #0.0>	Compare two floating-point registers, or one floating-point register and zero with Invalid Operation check	FPSCR
VCVT.S32.F32	Sd, Sm	Convert between floating-point and integer	–

**Table 1-12. Cortex-M4F Instruction Summary (continued)**

Mnemonic	Operands	Brief Description	Flags
VCVT.S16.F32	Sd, Sd, #fbits	Convert between floating-point and fixed point	–
VCVTR.S32.F32	Sd, Sm	Convert between floating-point and integer with rounding	–
VCVT<B H>.F32.F16	Sd, Sm	Converts half-precision value to single-precision	–
VCVTT<B T>.F32.F16	Sd, Sm	Converts single-precision register to half-precision	–
VDIV.F32	{Sd,} Sn, Sm	Floating-point Divide	–
VFMA.F32	{Sd,} Sn, Sm	Floating-point Fused Multiply Accumulate	–
VFNMA.F32	{Sd,} Sn, Sm	Floating-point Fused Negate Multiply Accumulate	–
VFMS.F32	{Sd,} Sn, Sm	Floating-point Fused Multiply Subtract	–
VFNMS.F32	{Sd,} Sn, Sm	Floating-point Fused Negate Multiply Subtract	–
VLDM.F<32 64>	Rn{!}, list	Load Multiple extension registers	–
VLDR.F<32 64>	<Dd Sd>, [Rn]	Load an extension register from memory	–
VLMA.F32	{Sd,} Sn, Sm	Floating-point Multiply Accumulate	–
VLMS.F32	{Sd,} Sn, Sm	Floating-point Multiply Subtract	–
VMOV.F32	Sd, #imm	Floating-point Move immediate	–
VMOV	Sd, Sm	Floating-point Move register	–
VMOV	Sn, Rt	Copy ARM core register to single precision	–
VMOV	Sm, Sm1, Rt, Rt2	Copy 2 ARM core registers to 2 single precision	–
VMOV	Dd[x], Rt	Copy ARM core register to scalar	–
VMOV	Rt, Dn[x]	Copy scalar to ARM core register	–
VMRS	Rt, FPSCR	Move FPSCR to ARM core register or APSR	N, Z, C, V
VMSR	FPSCR, Rt	Move to FPSCR from ARM Core register	FPSCR
VMUL.F32	{Sd,} Sn, Sm	Floating-point Multiply	–
VNEG.F32	Sd, Sm	Floating-point Negate	–
VNMLA.F32	Sd, Sn, Sm	Floating-point Multiply and Add	–
VNMLS.F32	Sd, Sn, Sm	Floating-point Multiply and Subtract	–
VNMUL	{Sd,} Sn, Sm	Floating-point Multiply	–
VPOP	list	Pop extension registers	–
VPUSH	list	Push extension registers	–
VSQRT.F32	Sd, Sm	Calculates floating-point Square Root	–
VSTM	Rn{!}, list	Floating-point register Store Multiple	–
VSTR.F<32 64>	Sd, [Rn]	Stores an extension register to memory	–
VSUB.F<32 64>	{Sd,} Sn, Sm	Floating-point Subtract	–
WFE	–	Wait For Event	–
WFI	–	Wait For Interrupt	–

## Cortex-M4F Peripherals

This chapter provides information on the implementation of the Cortex-M4F processor peripherals in MSP432P4xx devices.

Topic	Page
2.1 Cortex-M4F Peripherals Introduction .....	71
2.2 Functional Peripherals Description .....	71
2.3 Debug Peripherals Description .....	81
2.4 Functional Peripherals Registers .....	83
2.5 Debug Peripherals Registers .....	170

## 2.1 Cortex-M4F Peripherals Introduction

The following peripherals are described in this chapter:

- SysTick (see [Section 2.2.1](#))
 

Provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism.
- Nested Vectored Interrupt Controller (NVIC) (see [Section 2.2.2](#))
  - Facilitates low-latency exception and interrupt handling
  - Controls power management
  - Implements system control registers
- System Control Block (SCB) (see [Section 2.2.3](#))
 

Provides system implementation information and system control, including configuration, control, and reporting of system exceptions.
- Memory Protection Unit (MPU) (see [Section 2.2.4](#))
 

Supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.
- Floating-Point Unit (FPU) (see [Section 2.2.5](#))
 

Fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

[Table 2-1](#) shows the address map of the Private Peripheral Bus (PPB). Some peripheral register regions are split into two address regions, as indicated by two addresses listed.

**Table 2-1. Core Peripheral Register Regions**

Address	Core Peripheral	Section
0xE000_E010-0xE000_E01F	System Timer	<a href="#">Section 2.2.1</a>
0xE000_E100-0xE000_E4EF 0xE000_EF00-0xE000_EF03	Nested Vectored Interrupt Controller	<a href="#">Section 2.2.2</a>
0xE000_E008-0xE000_E00F 0xE000_ED00-0xE000_ED3F	System Control Block	<a href="#">Section 2.2.3</a>
0xE000_ED90-0xE000_EDB8	Memory Protection Unit	<a href="#">Section 2.2.4</a>
0xE000_EF30-0xE000_EF44	Floating Point Unit	<a href="#">Section 2.2.5</a>

## 2.2 Functional Peripherals Description

This chapter describes the implementation of the Cortex-M4 processor functional peripherals: SysTick, NVIC, SCB, MPU, FPU.

### 2.2.1 System Timer (SysTick)

Cortex-M4 includes an integrated system timer, SysTick, which provides a simple 24-bit clear-on-write decrementing wrap-on-zero counter with a flexible control mechanism. The counter can be used in several different ways, for example:

- An RTOS tick timer that fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the system clock.
- A variable rate alarm or signal timer—the duration is range-dependent on the reference clock used and the dynamic range of the counter.
- A simple counter used to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNT bit in the STCSR control and status register can be used to determine if an action completed within a set duration, as

part of a dynamic clock management control loop.

The timer consists of three registers. See [Section 2.4.4](#) for bits description of the SYSTICK registers:

- SysTick Control and Status Register (STCSR): A control and status register to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status.
- SysTick Reload Value Register (STRVR): The reload value for the counter, used to provide the counter's wrap value.
- SysTick Current Value Register (STCVR): The current value of the counter.

When enabled, the timer counts down on each clock from the reload value to zero, reloads (wraps) to the value in the STRVR register on the next clock edge, then decrements on subsequent clocks. Clearing the STRVR register disables the counter on the next wrap. When the counter reaches zero, the COUNT status bit is set. The COUNT bit clears on reads.

Writing to the STCVR register clears the register and the COUNT status bit. The write does not trigger the SysTick exception logic. On a read, the current value is the value of the register at the time the register is accessed.

The SysTick counter reload and current value are undefined at reset; the correct initialization sequence for the SysTick counter is:

1. Program the value in the STRVR register.
2. Clear the STCVR register by writing to it with any value.
3. Configure the STCSR register for the required operation.

---

**NOTE:** When the processor is halted for debugging, the counter does not decrement.

The timer is clocked with respect to a reference clock. The reference clock in MSP432P4xx devices is same as the CPU free running clock (FCLK). There is no support for an external clock to be used as SysTICK reference clock. Consequently, the CLKSOURCE bit in the SYSTICK Control and Status register must always be written as 1 (to indicate CPU clock as reference clock).

---

## 2.2.2 Nested Vectored Interrupt Controller (NVIC)

This section describes the Nested Vectored Interrupt Controller (NVIC) and the registers it uses. (See [Section 2.4.3](#) for bit description of NVIC registers).

The NVIC supports:

- 64 interrupts.
- A programmable priority level of 0-7 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Low-latency exception and interrupt handling.
- Level and pulse detection of interrupt signals.
- Dynamic re-prioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external Non-maskable interrupt (NMI).

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead, providing low latency exception handling.

### 2.2.2.1 Level-Sensitive and Pulse Interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt (see [Section 2.2.2.2](#) for more information). For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. As a result, the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

### 2.2.2.2 Hardware and Software Control of Interrupts

The Cortex-M4 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- The NVIC detects that the interrupt signal is High and the interrupt is not active.
- The NVIC detects a rising edge on the interrupt signal.
- Software writes to the corresponding interrupt set-pending register bit, or to the Software Trigger Interrupt Register (STIR) to make a Software-Generated Interrupt pending. See the INT bit in the IPSR0 register or the STIR register.

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt, changing the state of the interrupt from pending to active. Then:
  - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.
  - For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR.
- If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.
- Software writes to the corresponding interrupt clear-pending register bit
  - For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive.
  - For a pulse interrupt, the state of the interrupt changes to inactive, if the state was pending or to active, if the state was active and pending.

### 2.2.3 System Control Block (SCB)

The System Control Block (SCB) provides system implementation information and system control, including configuration, control, and reporting of the system exceptions. See [Section 2.4.5](#) for bit description of System Control Block registers.

## 2.2.4 Memory Protection Unit (MPU)

This section describes the Memory protection unit (MPU). The MPU divides the memory map into a number of regions and defines the location, size, access permissions, and memory attributes of each region. The MPU supports independent attribute settings for each region, overlapping regions, and export of memory attributes to the system.

The MPU provides full support for:

- Protection regions – eight distinct regions are supported
- Overlapping protection regions, with ascending region priority
  - 7 = highest priority
  - 0 = lowest priority
- Access permissions
- Exporting memory attributes to the system

MPU mismatches and permission violations invoke the programmable-priority MemManage fault handler. See the ARMv7-M Architecture Reference Manual ([ARM DDI0403E.B](#)) for more information.

You can use the MPU to:

- enforce privilege rules
- separate processes
- enforce access rules

The MPU registers are listed in [Section 2.4.2](#).

The access permission bits TEX, C, B, AP, and XN, of the Region Access Control Register controls access to the corresponding memory region. (See [Section 2.2.4.1](#) for more information on access permission bits). If an access is made to an area of memory without the required permissions, a permission fault is raised. If MemManage fault is enabled in the system, then it interrupts the processor as MemManage fault else it escalates itself as Hard Fault to interrupt the processor.

Each memory region of size above 256 bytes can be divided into 8 sub-regions of equal size and each of these sub-regions can be selectively disabled for protection via MPU.

When memory regions overlap, a memory access is affected by the attributes of the region with the highest number. For example, the attributes for region 7 take precedence over the attributes of any region that overlaps region 7.

The background region has the same memory access attributes as the default memory map, but is accessible from privileged software only.

The Cortex-M4 MPU memory map is unified, meaning that instruction accesses and data accesses have the same region settings.

If a program accesses a memory location that is prohibited by the MPU, the processor generates a memory management fault, causing a fault exception and possibly causing termination of the process in an OS environment. In an OS environment, the kernel can update the MPU region setting dynamically based on the process to be executed. Typically, an embedded OS uses the MPU for memory protection.

Configuration of MPU regions is based on memory types (see [Section 1.4.1](#) for more information on memory type definition ).

See [Section 2.2.4.1.1](#) for guidelines for programming the access permission bits of MPU in MSP432P4xx devices.

To avoid unexpected behavior, disable the interrupts before updating the attributes of a region that the interrupt handlers might access.

Ensure software uses aligned accesses of the correct size to access MPU registers:

- Except for the MPU Region Attribute and Size Register (RASR), all MPU registers must be accessed with aligned word accesses.
- The RASR register can be accessed with byte or aligned halfword or word accesses. The processor does not support unaligned accesses to MPU registers.

When setting up the MPU, and if the MPU has previously been programmed, disable unused regions to prevent any previous region settings from affecting the new MPU setup.

#### 2.2.4.1 MPU Access Permission Attributes

The access permission bits, TEX, S, C, B, AP, and XN of the RASR register, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

**Table 2-2** shows the encodings for the TEX, C, B, and S access permission bits. All encodings are shown for completeness, however the current implementation of the Cortex-M4 does not support the concept of cacheability or shareability. Refer to [Section 2.2.4.1.1](#) for information on programming the MPU.

**Table 2-2. TEX, S, C, and B Bit Field Encoding**

TEX	S	C	B	Memory type	Shareability	Other attributes
000b	x <sup>(1)</sup>	0	0	Strongly Ordered	Shareable	-
000b	x <sup>(1)</sup>	0	1	Device	Shareable	-
000b	0	1	0	Normal	Not shareable	Outer and inner write-through. No write allocate.
000b	1	1	0	Normal	Shareable	
000b	0	1	1	Normal	Not shareable	
000b	1	1	1	Normal	Shareable	
001b	0	0	0	Normal	Not shareable	
001b	1	0	0	Normal	Shareable	Outer and inner noncacheable.
001b	x <sup>(1)</sup>	0	1	Reserved encoding	-	
001b	x <sup>(1)</sup>	1	0	Reserved encoding	-	-
001b	0	1	1	Normal	Not shareable	Outer and inner write-back. Write and read allocate.
001b	1	1	1	Normal	Shareable	
010b	x <sup>(1)</sup>	0	0	Device	Not shareable	Nonshared Device.
010b	x <sup>(1)</sup>	0	1	Reserved encoding	-	-
010b	x <sup>(1)</sup>	1	x <sup>(1)</sup>	Reserved encoding	-	-
1BB	0	A	A	Normal	Not shareable	Cached memory , BB = outer policy, AA = inner policy. See <a href="#">Table 2-3</a> for the encoding of the AA and BB bits.
1BB	1	A	A	Normal	Shareable	

<sup>(1)</sup> The MPU ignores the value of this bit.

**Table 2-3** shows the cache policy for memory attribute encodings with a TEX value in the range of 0x4-0x7.

**Table 2-3. Cache Policy for Memory Attribute Encoding**

Encoding, AA or BB	Corresponding Cache Policy
00	Non-cacheable
01	Write back, write and read allocate
10	Write through, no write allocate
11	Write back, no write allocate

**Table 2-4** shows the AP encodings in the RASR register that define the access permissions for privileged and unprivileged software.

**Table 2-4. AP Bit Field Encoding**

AP Bit Field	Privileged Permissions	Unprivileged Permissions	Description
000	No access	No access	All accesses generate a permission fault.

**Table 2-4. AP Bit Field Encoding (continued)**

AP Bit Field	Privileged Permissions	Unprivileged Permissions	Description
001	RW	No access	Access from privileged software only.
010	RW	RO	Writes by unprivileged software generate a permission fault.
011	RW	RW	Full access.
100	Unpredictable	Unpredictable	Reserved
101	RO	No access	Reads by privileged software only.
110	RO	RO	Read-only, by privileged or unprivileged software.
111	RO	RO	Read-only, by privileged or unprivileged software.

#### 2.2.4.1.1 MPU Configuration for MSP432P4xx devices

MSP432P4xx has only a single processor and no caches. As a result, the MPU should be programmed as shown in [Table 2-5](#).

**Table 2-5. Memory Region Attributes for MSP432P4xx Devices**

Memory Region	TEX	S	C	B	Memory Type and Attributes
Flash memory	000b	0	1	0	Normal memory, non-shareable, write-through
Internal SRAM	000b	1	1	0	Normal memory, shareable, write-through
External SRAM	000b	1	1	1	Normal memory, shareable, write-back, write-allocate
Peripherals	000b	1	0	1	Device memory, shareable

In current MSP432P4xx family microcontroller implementations, the shareability and cache policy attributes do not affect the system behavior. However, using these settings for the MPU regions can make the application code more portable. The values given are for typical situations.

#### 2.2.4.2 Updating an MPU Region

To update the attributes for an MPU region, the MPU Region Number (RNR), MPU Region Base Address (RBAR), and RASR registers must be updated. Each register can be programmed separately or with a multiple-word write to program all of these registers. You can use the RBARx and RASRx aliases to program up to four regions simultaneously using an STM instruction.

##### Updating an MPU Region Using Separate Words

This example simple code configures one region:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=RNR           ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]    ; Region Number
STR R4, [R0, #0x4]    ; Region Base Address
STRH R2, [R0, #0x8]   ; Region Size and Enable
STRH R3, [R0, #0xA]   ; Region Attribute
```

Disable a region before writing new region settings to the MPU if you have previously enabled the region being changed. For example:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=RNR           ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]    ; Region Number
```

```

BIC R2, R2, #1      ; Disable
STRH R2, [R0, #0x8]  ; Region Size and Enable
STR R4, [R0, #0x4]   ; Region Base Address
STRH R3, [R0, #0xA]  ; Region Attribute
ORR R2, #1           ; Enable
STRH R2, [R0, #0x8]  ; Region Size and Enable

```

Software must use memory barrier instructions:

- Before MPU setup, if there might be outstanding memory transfers, such as buffered writes, that might be affected by the change in MPU settings.
- After MPU setup, if it includes memory transfers that must use the new MPU settings.

However, memory barrier instructions are not required if the MPU setup process starts by entering an exception handler, or is followed by an exception return, because the exception entry and exception return mechanism cause memory barrier behavior.

Software does not need any memory barrier instructions during MPU setup, because it accesses the MPU through the Private Peripheral Bus (PPB), which is a Strongly Ordered memory region.

For example, if all of the memory access behavior is intended to take effect immediately after the programming sequence, then a DSB instruction and an ISB instruction should be used. A DSB is required after changing MPU settings, such as at the end of context switch. An ISB is required if the code that programs the MPU region or regions is entered using a branch or call. If the programming sequence is entered using a return from exception, or by taking an exception, then an ISB is not required.

### **Updating an MPU Region Using Multi-Word Writes**

The MPU can be programmed directly using multi-word writes, depending how the information is divided. Consider the following reprogramming:

```

; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =RNR      ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0] ; Region Number
STR R2, [R0, #0x4] ; Region Base Address
STR R3, [R0, #0x8] ; Region Attribute, Size and Enable

```

An STM instruction can be used to optimize this:

```

; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =RNR      ; 0xE000ED98, MPU region number register
STM R0, {R1-R3}   ; Region number, address, attribute, size and enable

```

This operation can be done in two words for prepended information, meaning that the MPU Region Base Address (MPUBASE) register contains the required region number and has the VALID bit set. This method can be used when the data is statically packed, for example in a boot loader:

```

; R1 = address and region number in one
; R2 = size and attributes in one
LDR R0, = RBAR     ; 0xE000ED9C, MPU Region Base register
STR R1, [R0, #0x0] ; Region base address and region number combined
; with VALID (bit 4) set
; STR R2, [R0, #0x4] ; Region Attribute, Size and Enable

```

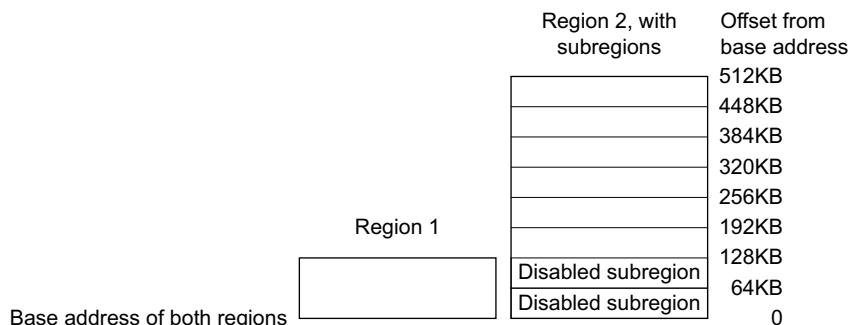
### **Subregions**

Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the SRD field of the MPU Region Base Address Register (RBAR) to disable a subregion. The least-significant bit of the SRD field controls the first subregion, and the most-significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion, the MPU issues a fault.

Regions of 32, 64, and 128 bytes do not support subregions. With regions of these sizes, the SRD field must be configured to 0x00, otherwise the MPU behavior is unpredictable.

### Example of SRD Use

Two regions with the same base address overlap. Region one is 128 KB, and region two is 512 KB. To ensure the attributes from region one apply to the first 128 KB region, configure the SRD field for region two to 0x03 to disable the first two subregions, as Figure 2-1 shows.



**Figure 2-1. SRD Use Example**

#### 2.2.4.3 MPU Mismatch

When an access violates the MPU permissions, the processor generates a memory management fault (see [Section 1.5.2](#) for more information). The MMFSR register indicates the cause of the fault.

### 2.2.5 Floating-Point Unit (FPU)

This section describes the Floating-Point Unit (FPU) and the registers it uses. The FPU provides:

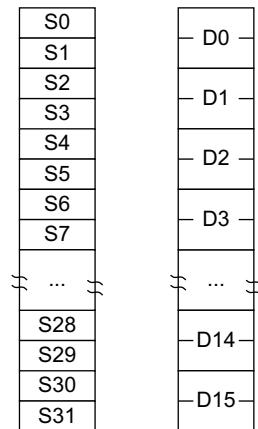
- 32-bit instructions for single-precision (C float) data-processing operations
- Combined multiply and accumulate instructions for increased precision (Fused MAC)
- Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square root
- Hardware support for denormals and all IEEE rounding modes
- 32 dedicated 32-bit single-precision registers, also addressable as 16 double-word registers
- Decoupled three stage pipeline

The Cortex-M4F FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions. The FPU provides floating-point computation functionality that is compliant with the ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard. The FPU's single-precision extension registers can also be accessed as 16 double-word registers for load, store, and move operations.

#### 2.2.5.1 FPU Views of the Register Bank

The FPU provides an extension register file containing 32 single-precision registers. These can be viewed as:

- Sixteen 64-bit double-word registers, D0-D15
- Thirty-two 32-bit single-word registers, S0-S31
- A combination of registers from the above views



**Figure 2-2. FPU Register Bank**

The mapping between the registers is as follows:

- $S<2n>$  maps to the least significant half of  $D<n>$
- $S<2n+1>$  maps to the most significant half of  $D<n>$

For example, you can access the least significant half of the value in D6 by accessing S12, and the most significant half of the elements by accessing S13.

#### 2.2.5.2 Modes of Operation

The control and status registers of FPU are listed in [Section 2.4.1](#).

The FPU provides three modes of operation to accommodate a variety of applications.

- Full-Compliance mode: In Full-Compliance mode, the FPU processes all operations according to the IEEE 754 standard in hardware.
- Flush-to-Zero mode: Setting the FZ bit of the Floating-Point Status and Control Register (FPSCR) enables Flush-to-Zero mode. In this mode, the FPU treats all subnormal input operands of arithmetic CDP operations as zeros in the operation. Exceptions that result from a zero operand are signalled appropriately. VABS, VNEG, and VMOV are not considered arithmetic CDP operations and are not affected by Flush-to-Zero mode. A result that is tiny, as described in the IEEE 754 standard, where the destination precision is smaller in magnitude than the minimum normal value before rounding, is replaced with a zero. The IDC bit in FPSCR indicates when an input flush occurs. The UFC bit in FPSCR indicates when a result flush occurs.
- Default NaN mode: Setting the DN bit in the FPSCR register enables default NaN mode. In this mode, the result of any arithmetic data processing operation that involves an input NaN, or that generates a NaN result, returns the default NaN. Propagation of the fraction bits is maintained only by VABS, VNEG, and VMOV operations. All other CDP operations ignore any information in the fraction bits of an input NaN.

#### 2.2.5.3 Compliance With the IEEE 754 Standard

When Default NaN (DN) and Flush-to-Zero (FZ) modes are disabled, FPv4 functionality is compliant with the IEEE 754 standard in hardware. No support code is required to achieve this compliance.

#### 2.2.5.4 Complete Implementation of the IEEE 754 Standard

The Cortex-M4F floating point instruction set does not support all operations defined in the IEEE 754-2008 standard. Unsupported operations include, but are not limited to the following:

- Remainder
- Round floating-point number to integer-valued floating-point number
- Binary-to-decimal conversions

- Decimal-to-binary conversions
- Direct comparison of single-precision and double-precision values

The Cortex-M4 FPU supports fused MAC operations as described in the IEEE standard. For complete implementation of the IEEE 754-2008 standard, floating-point functionality must be augmented with library functions.

### 2.2.5.5 IEEE 754 Standard Implementation Choices

#### NaN handling

All single-precision values with the maximum exponent field value and a nonzero fraction field are valid NaNs. A most-significant fraction bit of zero indicates a Signaling NaN (SNaN). A one indicates a Quiet NaN (QNaN). Two NaN values are treated as different NaNs if they differ in any bit. The following table shows the default NaN values.

Sign	Fraction	Fraction
0	0xFF	bit [22] = 1, bits [21:0] are all zeros

Processing of input NaNs for ARM floating-point functionality and libraries is defined as follows:

- In full-compliance mode, NaNs are handled as described in the ARM Architecture Reference Manual. The hardware processes the NaNs directly for arithmetic CDP instructions. For data transfer operations, NaNs are transferred without raising the Invalid Operation exception. For the non-arithmetic CDP instructions, VABS, VNEG, and VMOV, NaNs are copied, with a change of sign if specified in the instructions, without causing the Invalid Operation exception.
- In default NaN mode, arithmetic CDP instructions involving NaN operands return the default NaN regardless of the fractions of any NaN operands. SNaNs in an arithmetic CDP operation set the IOC flag, FPSCR[0]. NaN handling by data transfer and non-arithmetic CDP instructions is the same as in full-compliance mode.

**Table 2-6. QNaN and SNaN Handling**

Instruction Type	Default NaN Mode	With QNaN Operand	With SNaN Operand
Arithmetic CDP	Off	The QNaN or one of the QNaN operands, if there is more than one, is returned according to the rules given in the ARM Architecture Reference Manual.	IOC <sup>(1)</sup> set. The SNaN is quieted and the result NaN is determined by the rules given in the ARM Architecture Reference Manual.
	On	Default NaN returns.	IOC <sup>(1)</sup> set. Default NaN returns.
Non-arithmetic CDP	Off/On	NaN passes to destination with sign changed as appropriate.	
FCMP(Z)	-	Unordered compare	IOC set. Unordered compare.
FCMPE(Z)	-	IOC set. Unordered compare.	IOC set. Unordered compare.
Load/store	Off/On	All NaNs transferred.	

<sup>(1)</sup> IOC is the Invalid Operation exception flag, FPSCR[0].

#### Comparisons

Comparison results modify the flags in the FPSCR. You can use the MVRS APSR\_nzcv instruction (formerly FMSTAT) to transfer the current flags from the FPSCR to the APSR. See the ARM Architecture Reference Manual for mapping of IEEE 754-2008 standard predicates to ARM conditions. The flags used are chosen so that subsequent conditional execution of ARM instructions can test the predicates defined in the IEEE standard.

#### Underflow

The Cortex-M4F FPU uses the before rounding form of tininess and the inexact result form of loss of accuracy as described in the IEEE 754-2008 standard to generate Underflow exceptions.

In flush-to-zero mode, results that are tiny before rounding, as described in the IEEE standard, are flushed to a zero, and the UFC flag, FPSCR[3], is set. See the ARM Architecture Reference Manual for information on flush-to-zero mode.

When the FPU is not in flush-to-zero mode, operations are performed on subnormal operands. If the operation does not produce a tiny result, it returns the computed result, and the UFC flag, FPSCR[3], is not set. The IXC flag, FPSCR[4], is set if the operation is inexact. If the operation produces a tiny result, the result is a subnormal or zero value, and the UFC flag, FPSCR[3], is set if the result was also inexact.

### 2.2.5.6 Exceptions

The FPU sets the cumulative exception status flag in the FPSCR register as required for each instruction, in accordance with the FPv4 architecture. The FPU does not support user-mode traps. The exception enable bits in the FPSCR read-as-zero, and writes are ignored. The processor also has six output pins, FPIXC, FPUFC, FPOFC, FPDZC, FPIDC, and FPIOC, that each reflect the status of one of the cumulative exception flags. All these outputs are ORed and given on a single interrupt line in MSP432Pxx devices.

The processor can reduce the exception latency by using lazy stacking. See the Auxiliary Control Register (ACTLR). This means that the processor reserves space on the stack for the FP state, but does not save that state information to the stack. See the ARMv7-M Architecture Reference Manual (available from ARM) for more information.

### 2.2.5.7 Enabling the FPU

The FPU is disabled from reset. You must enable it before you can use any floating-point instructions. The processor must be in privileged mode to read from and write to the Coprocessor Access Control (CPAC) register. The following example code sequence enables the FPU in both privileged and user modes.

```
; CPACR is located at address 0xE000ED88
LDR.W R0, =0xE000ED88
; Read CPACR
LDR R1, [R0]
; Set bits 20-23 to enable CP10 and CP11 coprocessors
ORR R1, R1, #(0xF << 20)
; Write back the modified value to the CPACR
STR R1, [R0]; wait for store to complete
DSB
;reset pipeline now the FPU is enabled
ISB
```

## 2.3 Debug Peripherals Description

MSP432P4xx debug subsystem comprises the following components.

### 2.3.1 FPB

Flash Patch and Breakpoint unit supports six instruction comparators and two literal comparators for breakpoint. Code patching through FPB is supported in MSP432P4xx devices only when the device security features (JTAG and SWD lock or IP protection) are not enabled. When device security is enabled, any attempt to patch using FPB generates an internal reset to the system.

### 2.3.2 DWT

Data Watchpoint and Trace Unit containing four watchpoint units. It can be used for the following:

- A hardware watchpoint
- A PC sampler event trigger
- A data address sampler event trigger.

The DWT contains counters for:

- Clock cycles (CYCCNT)
- Folded instructions
- Load Store Unit (LSU) operations
- Sleep cycles
- CPI, that is all instruction cycles except for the first cycle

- Interrupt overhead

DWT can be configured to generate PC samples at defined intervals, and to generate interrupt event information. The DWT provides periodic requests for protocol synchronization to the ITM and the TPIU.

### 2.3.3 ITM

Instrumentation Trace Macrocell is an optional application-driven trace source that supports printf style debugging to trace operating system and application events, and generates diagnostic system information.

The ITM generates trace information as packets. There are various sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The different sources in decreasing order of priority are:

- Software trace: Software can write directly to ITM stimulus registers to generate packets.
- Hardware trace: The DWT generates these packets, and the ITM outputs them.
- Time stamping: Timestamps are generated relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex-M4 clock or the bitclock rate of the Serial Wire Viewer (SWV) output clocks the counter.

---

**NOTE:** ITM registers are fully accessible in privileged mode. In user mode, all registers can be read, but only the Stimulus registers and Trace Enable registers can be written, and only when the corresponding Trace Privilege register bit is set. Invalid user mode writes to the ITM registers are discarded.

---

### 2.3.4 TPIU

Trace Port Interface Unit is an optional component that acts as a bridge between the on-chip trace data from the Instrumentation Trace Macrocell (ITM) to a data stream. The TPIU encapsulates IDs where required, and the data stream is then captured by a Trace Port Analyzer (TPA).

The TPIU can output trace data in a Serial Wire Output (SWO) format.

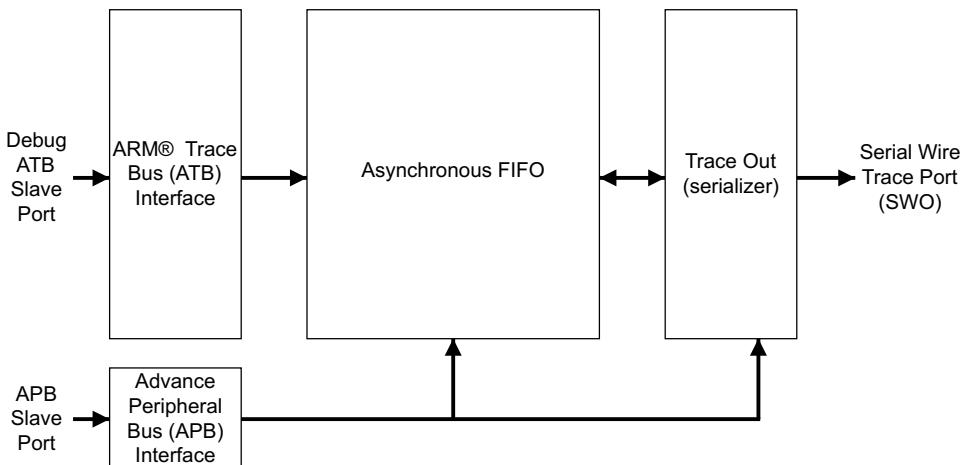


Figure 2-3. TPIU Block Diagram

#### 2.3.4.1 TPIU Components

- Asynchronous FIFO: The asynchronous FIFO enables trace data to be driven out at a speed that is not dependent on the speed of the core clock.
- Formatter: The formatter inserts source ID signals into the data packet stream so that trace data can be re-associated with its trace source. The formatter is always active when the TRACEPORT mode is active.
- Trace Out: The trace out block serializes formatted data before it goes off-chip. In MSP432P4xx

devices, a single trace pin (SWO) is available and TRACEDATA ports are not used.

---

**NOTE:** The SWO pin in MSP432P4xx devices is shared with the JTAG TDO pin and, therefore, JTAG and Trace features cannot be used together. To use trace features, Serial-Wire debug mode must be used.

---

## 2.4 Functional Peripherals Registers

This section lists the Cortex-M4 Peripheral SysTick, NVIC, MPU, FPU and SCB registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000\_E000.

---

**NOTE:** Register spaces that are not used are reserved for future or internal use. Software should not modify any reserved memory address.

---

### 2.4.1 FPU Registers

Table 2-7 lists the memory-mapped registers for the FPU. All register offset addresses not listed in Table 2-7 should be considered as reserved locations and the register contents should not be modified.

**Table 2-7. FPU Registers**

Offset	Acronym	Register Name	Type	Reset	Section
F34h	FPCCR	Floating Point Context Control Register	read-write	C0000000h	<a href="#">Section 2.4.1.1</a>
F38h	FPCAR	Floating-Point Context Address Register	read-write	00000000h	<a href="#">Section 2.4.1.2</a>
F3Ch	FPDSCR	Floating Point Default Status Control Register	read-write	00000000h	<a href="#">Section 2.4.1.3</a>
F40h	MVFR0	Media and FP Feature Register 0 (MVFR0)	read-only	10110021h	<a href="#">Section 2.4.1.4</a>
F44h	MVFR1	Media and FP Feature Register 1 (MVFR1)	read-only	11000011h	<a href="#">Section 2.4.1.5</a>

#### 2.4.1.1 FPCCR Register (Offset = F34h) [reset = C0000000h]

FPCCR is shown in [Figure 2-4](#) and described in [Table 2-8](#).

Floating Point Context Control Register. The Floating Point Context Control Register (FPCCR) holds control data for the floating-point unit. Accessible only by privileged software.

**Figure 2-4. FPCCR Register**

31	30	29	28	27	26	25	24
ASPEN	LSPEN			RESERVED			
R/W-1h	R/W-1h			R/W-0h			
23	22	21	20	19	18	17	16
			RESERVED				
			R/W-0h				
15	14	13	12	11	10	9	8
			RESERVED			MONRDY	
			R/W-0h			R/W-0h	
7	6	5	4	3	2	1	0
RESERVED	BFRDY	MMRDY	HFRDY	THREAD	RESERVED	USER	LSPACT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

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

Bit	Field	Type	Reset	Description
31	ASPEN	R/W	1h	Automatic State Preservation Enable. When this bit is set, it causes bit [2] of the Special CONTROL register to be set (FPCA) on execution of a floating point instruction, which results in the floating point state automatically being preserved on exception entry.
30	LSPEN	R/W	1h	Lazy State Preservation Enable. When the processor performs a context save, space on the stack is reserved for the floating point state, but it is not stacked until the new context performs a floating point operation.
29-9	RESERVED	R/W	0h	
8	MONRDY	R/W	0h	Indicates whether the software executing when the processor allocated the FP stack frame was able to set the DebugMonitor exception to pending.
7	RESERVED	R/W	0h	
6	BFRDY	R/W	0h	Indicates whether the software executing when the processor allocated the FP stack frame was able to set the BusFault exception to pending.
5	MMRDY	R/W	0h	Indicates whether the software executing when the processor allocated the FP stack frame was able to set the MemManage exception to pending.
4	HFRDY	R/W	0h	Indicates whether the software executing when the processor allocated the FP stack frame was able to set the HardFault exception to pending.
3	THREAD	R/W	0h	Indicates the processor mode was Thread when it allocated the FP stack frame.
2	RESERVED	R/W	0h	
1	USER	R/W	0h	Indicates the privilege level of the software executing was User (Unprivileged) when the processor allocated the FP stack frame.
0	LSPACT	R/W	0h	Indicates whether Lazy preservation of the FP state is active.

#### 2.4.1.2 FPCAR Register (Offset = F38h) [reset = 00000000h]

FPCAR is shown in [Figure 2-5](#) and described in [Table 2-9](#).

Floating-Point Context Address Register. The Floating-Point Context Address Register (FPCAR) holds the location of the unpopulated floating-point register space allocated on an exception stack frame. The FPCAR points to the stack location reserved for S0.

**Figure 2-5. FPCAR Register**

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

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

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	0h	
30-2	ADDRESS	R/W	0h	Holds the (double-word-aligned) location of the unpopulated floating-point register space allocated on an exception stack frame.
1-0	RESERVED	R/W	0h	

### 2.4.1.3 FPDSCR Register (Offset = F3Ch) [reset = 00000000h]

FPDSCR is shown in [Figure 2-6](#) and described in [Table 2-10](#).

Floating Point Default Status Control Register. The Floating Point Context Control Register (FPDSCR) holds the default values for the floating-point status control data that the processor assigns to the FPSCR when it creates a new floating-point context. Accessible only by privileged software.

**Figure 2-6. FPDSCR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
RESERVED		AHP	DN	FZ	RMODE		RESERVED									
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h		R/W-0h									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RESERVED															R/W-0h	

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

Bit	Field	Type	Reset	Description
31-27	RESERVED	R/W	0h	
26	AHP	R/W	0h	Default value for Alternative Half Precision bit. (If this bit is set to 1 then Alternative half-precision format is selected).
25	DN	R/W	0h	Default value for Default NaN mode bit. (If this bit is set to 1 then any operation involving one or more NaNs returns the Default NaN).
24	FZ	R/W	0h	Default value for Flush-to-Zero mode bit. (If this bit is set to 1 then Flush-to-zero mode is enabled).
23-22	RMODE	R/W	0h	Default value for Rounding Mode control field. (The encoding for this field is: 0b00 Round to Nearest (RN) mode, 0b01 Round towards Plus Infinity (RP) mode, 0b10 Round towards Minus Infinity (RM) mode, 0b11 Round towards Zero (RZ) mode. The specified rounding mode is used by almost all floating-point instructions).
21-0	RESERVED	R/W	0h	

#### 2.4.1.4 MVFR0 Register (Offset = F40h) [reset = 10110021h]

MVFR0 is shown in [Figure 2-7](#) and described in [Table 2-11](#).

Media and FP Feature Register 0 (MVFR0). Describes the features provided by the Floating-point extension.

**Figure 2-7. MVFR0 Register**

31	30	29	28	27	26	25	24
FP_ROUNDING_MODES				SHORT_VECTORS			
R-1h				R-0h			
23	22	21	20	19	18	17	16
SQUARE_ROOT				DIVIDE			
R-1h				R-1h			
15	14	13	12	11	10	9	8
FP_ECEPTION_TRAPPING				DOUBLE_PRECISION			
R-0h				R-0h			
7	6	5	4	3	2	1	0
SINGLE_PRECISION				A SIMD_REGISTERS			
R-2h				R-1h			

**Table 2-11. MVFR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	FP_ROUNDING_MODES	R	1h	Indicates the rounding modes supported by the FP floating-point hardware. The value of this field is: 0b0001 - all rounding modes supported.
27-24	SHORT_VECTORS	R	0h	Indicates the hardware support for FP short vectors. The value of this field is: 0b0000 - not supported in ARMv7-M.
23-20	SQUARE_ROOT	R	1h	Indicates the hardware support for FP square root operations. The value of this field is: 0b0001 - supported.
19-16	DIVIDE	R	1h	Indicates the hardware support for FP divide operations. The value of this field is: 0b0001 - supported.
15-12	FP_ECEPTION_TRAPPING	R	0h	Indicates whether the FP hardware implementation supports exception trapping. The value of this field is: 0b0000 - not supported in ARMv7-M.
11-8	DOUBLE_PRECISION	R	0h	Indicates the hardware support for FP double-precision operations. The value of this field is: 0b0000 - not supported in ARMv7-M.
7-4	SINGLE_PRECISION	R	2h	Indicates the hardware support for FP single-precision operations. The value of this field is: 0b0010 - supported.
3-0	A SIMD_REGISTERS	R	1h	Indicates the size of the FP register bank. The value of this field is: 0b0001 - supported, 16 x 64-bit registers.

#### 2.4.1.5 MVFR1 Register (Offset = F44h) [reset = 11000011h]

MVFR1 is shown in [Figure 2-8](#) and described in [Table 2-12](#).

Media and FP Feature Register 1 (MVFR1). Describes the features provided by the Floating-point extension.

**Figure 2-8. MVFR1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FP_FUSED_MAC				FP_HPFP				RESERVED							
R-1h				R-1h				R-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								D_NAN_MODE				FTZ_MODE			
R-0h								R-1h				R-1h			

**Table 2-12. MVFR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	FP_FUSED_MAC	R	1h	Indicates whether the FP supports fused multiply accumulate operations. The value of this field is: 0b0001 - supported.
27-24	FP_HPFP	R	1h	Indicates whether the FP supports half-precision floating-point conversion operations. The value of this field is: 0b0001 - supported.
23-8	RESERVED	R	0h	
7-4	D_NAN_MODE	R	1h	Indicates whether the FP hardware implementation supports only the Default NaN mode. The value of this field is: 0b0001 - hardware supports propagation of NaN values.
3-0	FTZ_MODE	R	1h	Indicates whether the FP hardware implementation supports only the Flush-to-Zero mode of operation. The value of this field is: 0b0001 - hardware supports full denormalized number arithmetic.

## 2.4.2 MPU Registers

Table 2-13 lists the memory-mapped registers for the MPU. All register offset addresses not listed in Table 2-13 should be considered as reserved locations and the register contents should not be modified.

**Table 2-13. MPU Registers**

Offset	Acronym	Register Name	Type	Reset	Section
D90h	TYPE	MPU Type Register	read-only	00000800h	Section 2.4.2.1
D94h	CTRL	MPU Control Register	read-write	00000000h	Section 2.4.2.2
D98h	RNR	MPU Region Number Register	read-write	00000000h	Section 2.4.2.3
D9Ch	RBAR	MPU Region Base Address Register	read-write	00000000h	Section 2.4.2.4
DA0h	RASR	MPU Region Attribute and Size Register	read-write	00000000h	Section 2.4.2.5
DA4h	RBAR_A1	MPU Alias 1 Region Base Address register	read-write	00000000h	Section 2.4.2.6
DA8h	RASR_A1	MPU Alias 1 Region Attribute and Size register	read-write	00000000h	Section 2.4.2.7
DACh	RBAR_A2	MPU Alias 2 Region Base Address register	read-write	00000000h	Section 2.4.2.8
DB0h	RASR_A2	MPU Alias 2 Region Attribute and Size register	read-write	00000000h	Section 2.4.2.9
DB4h	RBAR_A3	MPU Alias 3 Region Base Address register	read-write	00000000h	Section 2.4.2.10
DB8h	RASR_A3	MPU Alias 3 Region Attribute and Size register	read-write	00000000h	Section 2.4.2.11

#### 2.4.2.1 TYPE Register (Offset = D90h) [reset = 00000800h]

TYPE is shown in [Figure 2-9](#) and described in [Table 2-14](#).

MPU Type Register. Use the MPU Type Register to see how many regions the MPU supports. Read bits [15:8] to determine if an MPU is present.

**Figure 2-9. TYPE Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
IREGION							
R-0h							
15	14	13	12	11	10	9	8
DREGION							
R-8h							
7	6	5	4	3	2	1	0
RESERVED							SEPARATE
							R-0h

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

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	
23-16	IREGION	R	0h	Because the processor core uses only a unified MPU, IREGION always contains 0x00.
15-8	DREGION	R	8h	Number of supported MPU regions field. DREGION contains 0x08 if the implementation contains an MPU indicating eight MPU regions, otherwise it contains 0x00.
7-1	RESERVED	R	0h	
0	SEPARATE	R	0h	Because the processor core uses only a unified MPU, SEPARATE is always 0.

#### 2.4.2.2 CTRL Register (Offset = D94h) [reset = 00000000h]

CTRL is shown in [Figure 2-10](#) and described in [Table 2-15](#).

MPU Control Register. Use the MPU Control Register to enable the MPU, enable the default memory map (background region), and enable the MPU when in Hard Fault, Non-maskable Interrupt (NMI), and FAULTMASK escalated handlers. When the MPU is enabled, at least one region of the memory map must be enabled for the MPU to function unless the PRIVDEFENA bit is set. If the PRIVDEFENA bit is set and no regions are enabled, then only privileged code can operate. When the MPU is disabled, the default address map is used, as if no MPU is present. When the MPU is enabled, only the system partition and vector table loads are always accessible. Other areas are accessible based on regions and whether PRIVDEFENA is enabled. Unless HFNMIEA is set, the MPU is not enabled when the exception priority is -1 or -2. These priorities are only possible when in Hard fault, NMI, or when FAULTMASK is enabled. The HFNMIEA bit enables the MPU when operating with these two priorities.

**Figure 2-10. CTRL Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED					PRIVDEFENA	HFNMIEA	ENABLE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

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

Bit	Field	Type	Reset	Description
31-3	RESERVED	R/W	0h	
2	PRIVDEFENA	R/W	0h	This bit enables the default memory map for privileged access, as a background region, when the MPU is enabled. The background region acts as if it was region number 1 before any settable regions. Any region that is set up overlays this default map, and overrides it. If this bit = 0, the default memory map is disabled, and memory not covered by a region faults. This applies to memory type, Execute Never (XN), cache and shareable rules. However, this only applies to privileged mode (fetch and data access). User mode code faults unless a region has been set up for its code and data. When the MPU is disabled, the default map acts on both privileged and user mode code. XN and SO rules always apply to the System partition whether this enable is set or not. If the MPU is disabled, this bit is ignored. Reset clears the PRIVDEFENA bit.
1	HFNMIEA	R/W	0h	This bit enables the MPU when in Hard Fault, NMI, and FAULTMASK escalated handlers. If this bit = 1 and the ENABLE bit = 1, the MPU is enabled when in these handlers. If this bit = 0, the MPU is disabled when in these handlers, regardless of the value of ENABLE. If this bit =1 and ENABLE = 0, behavior is Unpredictable. Reset clears the HFNMIEA bit.
0	ENABLE	R/W	0h	MPU enable bit. Reset clears the ENABLE bit. 0b = disable MPU 1b = enable MPU

### 2.4.2.3 RNR Register (Offset = D98h) [reset = 0h]

RNR is shown in [Figure 2-11](#) and described in [Table 2-16](#).

MPU Region Number Register. Use the MPU Region Number Register to select which protection region is accessed. Then write to the MPU Region Base Address Register or the MPU Attributes and Size Register to configure the characteristics of the protection region.

**Figure 2-11. RNR Register**

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
RESERVED																										REGION					
R/W-																										R/W-					

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

Bit	Field	Type	Reset	Description
31-8	RESERVED	R/W		
7-0	REGION	R/W		Region select field. Selects the region to operate on when using the Region Attribute and Size Register and the Region Base Address Register. It must be written first except when the address VALID + REGION fields are written, which overwrites this.

#### 2.4.2.4 RBAR Register (Offset = D9Ch) [reset = 00000000h]

RBAR is shown in [Figure 2-12](#) and described in [Table 2-17](#).

MPU Region Base Address Register. Use the MPU Region Base Address Register to write the base address of a region. The Region Base Address Register also contains a REGION field that you can use to override the REGION field in the MPU Region Number Register, if the VALID bit is set. The Region Base Address register sets the base for the region. It is aligned by the size. So, a 64-KB sized region must be aligned on a multiple of 64KB, for example, 0x00010000 or 0x00020000. The region always reads back as the current MPU region number. VALID always reads back as 0. Writing with VALID = 1 and REGION = n changes the region number to n. This is a short-hand way to write the MPU Region Number Register. This register is Unpredictable if accessed other than as a word.

**Figure 2-12. RBAR Register**

31	30	29	28	27	26	25	24
ADDR							
R/W-0h							
23	22	21	20	19	18	17	16
ADDR							
R/W-0h							
15	14	13	12	11	10	9	8
ADDR							
R/W-0h							
7	6	5	4	3	2	1	0
ADDR			VALID		REGION		
R/W-0h			R/W-0h		R/W-0h		

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

Bit	Field	Type	Reset	Description
31-5	ADDR	R/W	0h	Region base address field. The position of the LSB depends on the region size, so that the base address is aligned according to an even multiple of size. The power of 2 size specified by the SZENABLE field of the MPU Region Attribute and Size Register defines how many bits of base address are used.
4	VALID	R/W	0h	MPU Region Number valid bit. 0b = MPU Region Number Register remains unchanged and is interpreted. 1b = MPU Region Number Register is overwritten by bits 3:0 (the REGION value).
3-0	REGION	R/W	0h	MPU region override field.

### 2.4.2.5 RASR Register (Offset = DA0h) [reset = 00000000h]

RASR is shown in [Figure 2-13](#) and described in [Table 2-18](#).

MPU Region Attribute and Size Register. Use the MPU Region Attribute and Size Register to control the MPU access permissions. The register is made up of two part registers, each of halfword size. These can be accessed using the individual size, or they can both be simultaneously accessed using a word operation. The sub-region disable bits are not supported for region sizes of 32 bytes, 64 bytes, and 128 bytes. When these region sizes are used, the subregion disable bits must be programmed as 0.

**Figure 2-13. RASR Register**

31	30	29	28	27	26	25	24
RESERVED		XN	RESERVED		AP		
R/W-0h		R/W-0h	R/W-0h		R/W-0h		
23	22	21	20	19	18	17	16
RESERVED		TEX		S	C	B	
R/W-0h		R/W-0h		R/W-0h	R/W-0h	R/W-0h	
15	14	13	12	11	10	9	8
SRD							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		SIZE				ENABLE	
R/W-0h		R/W-0h				R/W-0h	

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

Bit	Field	Type	Reset	Description
31-29	RESERVED	R/W	0h	
28	XN	R/W	0h	Instruction access disable bit 0b = enable instruction fetches 1b = disable instruction fetches
27	RESERVED	R/W	0h	
26-24	AP	R/W	0h	Data access permission field 0b = Privileged permissions: No access. User permissions: No access. 1b = Privileged permissions: Read-write. User permissions: No access. 10b = Privileged permissions: Read-write. User permissions: Read-only. 11b = Privileged permissions: Read-write. User permissions: Read-write. 101b = Privileged permissions: Read-only. User permissions: No access. 110b = Privileged permissions: Read-only. User permissions: Read-only. 111b = Privileged permissions: Read-only. User permissions: Read-only.
23-22	RESERVED	R/W	0h	
21-19	TEX	R/W	0h	Type extension field
18	S	R/W	0h	Shareable bit 0b = not shareable 1b = shareable
17	C	R/W	0h	Cacheable bit 0b = not cacheable 1b = cacheable

**Table 2-18. RASR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	B	R/W	0h	Bufferable bit 0b = not bufferable 1b = bufferable
15-8	SRD	R/W	0h	Sub-Region Disable (SRD) field. Setting an SRD bit disables the corresponding sub-region. Regions are split into eight equal-sized sub-regions. Sub-regions are not supported for region sizes of 128 bytes and less.
7-6	RESERVED	R/W	0h	
5-1	SIZE	R/W	0h	MPU Protection Region Size Field. 0b = Reserved 1b = Reserved 10b = Reserved 11b = Reserved 100b = 32B 101b = 64B 110b = 128B 111b = 256B 1000b = 512B 1001b = 1KB 1010b = 2KB 1011b = 4KB 1100b = 8KB 1101b = 16KB 1110b = 32KB 1111b = 64KB 10000b = 128KB 10001b = 256KB 10010b = 512KB 10011b = 1MB 10100b = 2MB 10101b = 4MB 10110b = 8MB 10111b = 16MB 11000b = 32MB 11001b = 64MB 11010b = 128MB 11011b = 256MB 11100b = 512MB 11101b = 1GB 11110b = 2GB 11111b = 4GB
0	ENABLE	R/W	0h	Region enable bit.

#### 2.4.2.6 RBAR\_A1 Register (Offset = DA4h) [reset = 00000000h]

RBAR\_A1 is shown in [Figure 2-14](#) and described in [Table 2-19](#).

MPU Alias 1 Region Base Address register. Alias of 0xE000ED9C.

**Figure 2-14. RBAR\_A1 Register**

31	30	29	28	27	26	25	24
ADDR							
R/W-0h							
23	22	21	20	19	18	17	16
ADDR							
R/W-0h							
15	14	13	12	11	10	9	8
ADDR							
R/W-0h							
7	6	5	4	3	2	1	0
ADDR			VALID		REGION		
R/W-0h			R/W-0h		R/W-0h		

**Table 2-19. RBAR\_A1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	ADDR	R/W	0h	Region base address field. The position of the LSB depends on the region size, so that the base address is aligned according to an even multiple of size. The power of 2 size specified by the SZENABLE field of the MPU Region Attribute and Size Register defines how many bits of base address are used.
4	VALID	R/W	0h	MPU Region Number valid bit. 0b = MPU Region Number Register remains unchanged and is interpreted. 1b = MPU Region Number Register is overwritten by bits 3:0 (the REGION value).
3-0	REGION	R/W	0h	MPU region override field.

#### 2.4.2.7 RASR\_A1 Register (Offset = DA8h) [reset = 00000000h]

RASR\_A1 is shown in [Figure 2-15](#) and described in [Table 2-20](#).

MPU Alias 1 Region Attribute and Size register. Alias of 0xE000EDA0.

**Figure 2-15. RASR\_A1 Register**

31	30	29	28	27	26	25	24
RESERVED			XN	RESERVED		AP	
R/W-0h			R/W-0h	R/W-0h			R/W-0h
23	22	21	20	19	18	17	16
RESERVED		TEX			S	C	B
R/W-0h			R/W-0h	R/W-0h		R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
SRD							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		SIZE				ENABLE	
R/W-0h		R/W-0h				R/W-0h	

**Table 2-20. RASR\_A1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R/W	0h	
28	XN	R/W	0h	Instruction access disable bit 0b = enable instruction fetches 1b = disable instruction fetches
27	RESERVED	R/W	0h	
26-24	AP	R/W	0h	Data access permission field 0b = Privileged permissions: No access. User permissions: No access. 1b = Privileged permissions: Read-write. User permissions: No access. 10b = Privileged permissions: Read-write. User permissions: Read-only. 11b = Privileged permissions: Read-write. User permissions: Read-write. 101b = Privileged permissions: Read-only. User permissions: No access. 110b = Privileged permissions: Read-only. User permissions: Read-only. 111b = Privileged permissions: Read-only. User permissions: Read-only.
23-22	RESERVED	R/W	0h	
21-19	TEX	R/W	0h	Type extension field
18	S	R/W	0h	Shareable bit 0b = not shareable 1b = shareable
17	C	R/W	0h	Cacheable bit 0b = not cacheable 1b = cacheable
16	B	R/W	0h	Bufferable bit 0b = not bufferable 1b = bufferable

**Table 2-20. RASR\_A1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15-8	SRD	R/W	0h	Sub-Region Disable (SRD) field. Setting an SRD bit disables the corresponding sub-region. Regions are split into eight equal-sized sub-regions. Sub-regions are not supported for region sizes of 128 bytes and less.
7-6	RESERVED	R/W	0h	
5-1	SIZE	R/W	0h	MPU Protection Region Size Field. 0b = Reserved 1b = Reserved 10b = Reserved 11b = Reserved 100b = 32B 101b = 64B 110b = 128B 111b = 256B 1000b = 512B 1001b = 1KB 1010b = 2KB 1011b = 4KB 1100b = 8KB 1101b = 16KB 1110b = 32KB 1111b = 64KB 10000b = 128KB 10001b = 256KB 10010b = 512KB 10011b = 1MB 10100b = 2MB 10101b = 4MB 10110b = 8MB 10111b = 16MB 11000b = 32MB 11001b = 64MB 11010b = 128MB 11011b = 256MB 11100b = 512MB 11101b = 1GB 11110b = 2GB 11111b = 4GB
0	ENABLE	R/W	0h	Region enable bit.

#### 2.4.2.8 RBAR\_A2 Register (Offset = DACH) [reset = 00000000h]

RBAR\_A2 is shown in [Figure 2-16](#) and described in [Table 2-21](#).

MPU Alias 2 Region Base Address register. Alias of 0xE000ED9C.

**Figure 2-16. RBAR\_A2 Register**

31	30	29	28	27	26	25	24
ADDR							
R/W-0h							
23	22	21	20	19	18	17	16
ADDR							
R/W-0h							
15	14	13	12	11	10	9	8
ADDR							
R/W-0h							
7	6	5	4	3	2	1	0
ADDR			VALID		REGION		
R/W-0h			R/W-0h		R/W-0h		

**Table 2-21. RBAR\_A2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	ADDR	R/W	0h	Region base address field. The position of the LSB depends on the region size, so that the base address is aligned according to an even multiple of size. The power of 2 size specified by the SZENABLE field of the MPU Region Attribute and Size Register defines how many bits of base address are used.
4	VALID	R/W	0h	MPU Region Number valid bit. 0b = MPU Region Number Register remains unchanged and is interpreted. 1b = MPU Region Number Register is overwritten by bits 3:0 (the REGION value).
3-0	REGION	R/W	0h	MPU region override field.

### 2.4.2.9 RASR\_A2 Register (Offset = DB0h) [reset = 00000000h]

RASR\_A2 is shown in [Figure 2-17](#) and described in [Table 2-22](#).

MPU Alias 2 Region Attribute and Size register. Alias of 0xE000EDA0.

**Figure 2-17. RASR\_A2 Register**

31	30	29	28	27	26	25	24
RESERVED			XN	RESERVED		AP	
R/W-0h			R/W-0h	R/W-0h			R/W-0h
23	22	21	20	19	18	17	16
RESERVED		TEX			S	C	B
R/W-0h			R/W-0h	R/W-0h		R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
SRD							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		SIZE				ENABLE	
R/W-0h			R/W-0h				R/W-0h

**Table 2-22. RASR\_A2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R/W	0h	
28	XN	R/W	0h	Instruction access disable bit 0b = enable instruction fetches 1b = disable instruction fetches
27	RESERVED	R/W	0h	
26-24	AP	R/W	0h	Data access permission field 0b = Privileged permissions: No access. User permissions: No access. 1b = Privileged permissions: Read-write. User permissions: No access. 10b = Privileged permissions: Read-write. User permissions: Read-only. 11b = Privileged permissions: Read-write. User permissions: Read-write. 101b = Privileged permissions: Read-only. User permissions: No access. 110b = Privileged permissions: Read-only. User permissions: Read-only. 111b = Privileged permissions: Read-only. User permissions: Read-only.
23-22	RESERVED	R/W	0h	
21-19	TEX	R/W	0h	Type extension field
18	S	R/W	0h	Shareable bit 0b = not shareable 1b = shareable
17	C	R/W	0h	Cacheable bit 0b = not cacheable 1b = cacheable
16	B	R/W	0h	Bufferable bit 0b = not bufferable 1b = bufferable

**Table 2-22. RASR\_A2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15-8	SRD	R/W	0h	Sub-Region Disable (SRD) field. Setting an SRD bit disables the corresponding sub-region. Regions are split into eight equal-sized sub-regions. Sub-regions are not supported for region sizes of 128 bytes and less.
7-6	RESERVED	R/W	0h	
5-1	SIZE	R/W	0h	MPU Protection Region Size Field. 0b = Reserved 1b = Reserved 10b = Reserved 11b = Reserved 100b = 32B 101b = 64B 110b = 128B 111b = 256B 1000b = 512B 1001b = 1KB 1010b = 2KB 1011b = 4KB 1100b = 8KB 1101b = 16KB 1110b = 32KB 1111b = 64KB 10000b = 128KB 10001b = 256KB 10010b = 512KB 10011b = 1MB 10100b = 2MB 10101b = 4MB 10110b = 8MB 10111b = 16MB 11000b = 32MB 11001b = 64MB 11010b = 128MB 11011b = 256MB 11100b = 512MB 11101b = 1GB 11110b = 2GB 11111b = 4GB
0	ENABLE	R/W	0h	Region enable bit.

### 2.4.2.10 RBAR\_A3 Register (Offset = DB4h) [reset = 00000000h]

RBAR\_A3 is shown in [Figure 2-18](#) and described in [Table 2-23](#).

MPU Alias 3 Region Base Address register. Alias of 0xE000ED9C.

**Figure 2-18. RBAR\_A3 Register**

31	30	29	28	27	26	25	24
ADDR							
R/W-0h							
23	22	21	20	19	18	17	16
ADDR							
R/W-0h							
15	14	13	12	11	10	9	8
ADDR							
R/W-0h							
7	6	5	4	3	2	1	0
ADDR			VALID		REGION		
R/W-0h			R/W-0h		R/W-0h		

**Table 2-23. RBAR\_A3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	ADDR	R/W	0h	Region base address field. The position of the LSB depends on the region size, so that the base address is aligned according to an even multiple of size. The power of 2 size specified by the SZENABLE field of the MPU Region Attribute and Size Register defines how many bits of base address are used.
4	VALID	R/W	0h	MPU Region Number valid bit. 0b = MPU Region Number Register remains unchanged and is interpreted. 1b = MPU Region Number Register is overwritten by bits 3:0 (the REGION value).
3-0	REGION	R/W	0h	MPU region override field.

#### 2.4.2.11 RASR\_A3 Register (Offset = DB8h) [reset = 00000000h]

RASR\_A3 is shown in [Figure 2-19](#) and described in [Table 2-24](#).

MPU Alias 3 Region Attribute and Size register. Alias of 0xE000EDA0.

**Figure 2-19. RASR\_A3 Register**

31	30	29	28	27	26	25	24
RESERVED			XN	RESERVED		AP	
R/W-0h			R/W-0h	R/W-0h			R/W-0h
23	22	21	20	19	18	17	16
RESERVED		TEX			S	C	B
R/W-0h			R/W-0h	R/W-0h		R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
SRD							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		SIZE				ENABLE	
R/W-0h		R/W-0h				R/W-0h	

**Table 2-24. RASR\_A3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R/W	0h	
28	XN	R/W	0h	Instruction access disable bit 0b = enable instruction fetches 1b = disable instruction fetches
27	RESERVED	R/W	0h	
26-24	AP	R/W	0h	Data access permission field 0b = Privileged permissions: No access. User permissions: No access. 1b = Privileged permissions: Read-write. User permissions: No access. 10b = Privileged permissions: Read-write. User permissions: Read-only. 11b = Privileged permissions: Read-write. User permissions: Read-write. 101b = Privileged permissions: Read-only. User permissions: No access. 110b = Privileged permissions: Read-only. User permissions: Read-only. 111b = Privileged permissions: Read-only. User permissions: Read-only.
23-22	RESERVED	R/W	0h	
21-19	TEX	R/W	0h	Type extension field
18	S	R/W	0h	Shareable bit 0b = not shareable 1b = shareable
17	C	R/W	0h	Cacheable bit 0b = not cacheable 1b = cacheable
16	B	R/W	0h	Bufferable bit 0b = not bufferable 1b = bufferable

**Table 2-24. RASR\_A3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
15-8	SRD	R/W	0h	Sub-Region Disable (SRD) field. Setting an SRD bit disables the corresponding sub-region. Regions are split into eight equal-sized sub-regions. Sub-regions are not supported for region sizes of 128 bytes and less.
7-6	RESERVED	R/W	0h	
5-1	SIZE	R/W	0h	MPU Protection Region Size Field. 0b = Reserved 1b = Reserved 10b = Reserved 11b = Reserved 100b = 32B 101b = 64B 110b = 128B 111b = 256B 1000b = 512B 1001b = 1KB 1010b = 2KB 1011b = 4KB 1100b = 8KB 1101b = 16KB 1110b = 32KB 1111b = 64KB 10000b = 128KB 10001b = 256KB 10010b = 512KB 10011b = 1MB 10100b = 2MB 10101b = 4MB 10110b = 8MB 10111b = 16MB 11000b = 32MB 11001b = 64MB 11010b = 128MB 11011b = 256MB 11100b = 512MB 11101b = 1GB 11110b = 2GB 11111b = 4GB
0	ENABLE	R/W	0h	Region enable bit.

### 2.4.3 NVIC Registers

Table 2-25 lists the memory-mapped registers for the NVIC. All register offset addresses not listed in Table 2-25 should be considered as reserved locations and the register contents should not be modified.

**Table 2-25. NVIC Registers**

Offset	Acronym	Register Name	Type	Reset	Section
100h	ISER0	Irq 0 to 31 Set Enable Register	read-write	00000000h	<a href="#">Section 2.4.3.1</a>
104h	ISER1	Irq 32 to 63 Set Enable Register	read-write	00000000h	<a href="#">Section 2.4.3.2</a>
180h	ICER0	Irq 0 to 31 Clear Enable Register	read-write	00000000h	<a href="#">Section 2.4.3.3</a>
184h	ICER1	Irq 32 to 63 Clear Enable Register	read-write	00000000h	<a href="#">Section 2.4.3.4</a>
200h	ISPR0	Irq 0 to 31 Set Pending Register	read-write	00000000h	<a href="#">Section 2.4.3.5</a>
204h	ISPR1	Irq 32 to 63 Set Pending Register	read-write	00000000h	<a href="#">Section 2.4.3.6</a>
280h	ICPR0	Irq 0 to 31 Clear Pending Register	read-write	00000000h	<a href="#">Section 2.4.3.7</a>
284h	ICPR1	Irq 32 to 63 Clear Pending Register	read-write	00000000h	<a href="#">Section 2.4.3.8</a>
300h	IABR0	Irq 0 to 31 Active Bit Register	read-only	00000000h	<a href="#">Section 2.4.3.9</a>
304h	IABR1	Irq 32 to 63 Active Bit Register	read-only	00000000h	<a href="#">Section 2.4.3.10</a>
400h	IPR0	Irq 0 to 3 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.11</a>
404h	IPR1	Irq 4 to 7 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.12</a>
408h	IPR2	Irq 8 to 11 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.13</a>
40Ch	IPR3	Irq 12 to 15 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.14</a>
410h	IPR4	Irq 16 to 19 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.15</a>
414h	IPR5	Irq 20 to 23 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.16</a>
418h	IPR6	Irq 24 to 27 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.17</a>
41Ch	IPR7	Irq 28 to 31 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.18</a>
420h	IPR8	Irq 32 to 35 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.19</a>
424h	IPR9	Irq 36 to 39 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.20</a>
428h	IPR10	Irq 40 to 43 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.21</a>
42Ch	IPR11	Irq 44 to 47 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.22</a>
430h	IPR12	Irq 48 to 51 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.23</a>
434h	IPR13	Irq 52 to 55 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.24</a>
438h	IPR14	Irq 56 to 59 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.25</a>
43Ch	IPR15	Irq 60 to 63 Priority Register	read-write	00000000h	<a href="#">Section 2.4.3.26</a>
F00h	STIR	Software Trigger Interrupt Register	write-only	00000000h	<a href="#">Section 2.4.3.27</a>

#### 2.4.3.1 ISER0 Register (Offset = 100h) [reset = 00000000h]

ISER0 is shown in [Figure 2-20](#) and described in [Table 2-26](#).

Irq 0 to 31 Set Enable Register. Use the Interrupt Set-Enable Registers to enable interrupts and determine which interrupts are currently enabled.

**Figure 2-20. ISER0 Register**

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
SETENA																															
R/W-0h																															

**Table 2-26. ISER0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SETENA	R/W	0h	Writing 0 to a SETENA bit has no effect, writing 1 to a bit enables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the SETENA fields.

#### 2.4.3.2 ISER1 Register (Offset = 104h) [reset = 00000000h]

ISER1 is shown in [Figure 2-21](#) and described in [Table 2-27](#).

Irq 32 to 63 Set Enable Register. Use the Interrupt Set-Enable Registers to enable interrupts and determine which interrupts are currently enabled.

**Figure 2-21. ISER1 Register**

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
SETENA																															
R/W-0h																															

**Table 2-27. ISER1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SETENA	R/W	0h	Writing 0 to a SETENA bit has no effect, writing 1 to a bit enables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the SETENA fields.

### 2.4.3.3 ICER0 Register (Offset = 180h) [reset = 00000000h]

ICER0 is shown in [Figure 2-22](#) and described in [Table 2-28](#).

Irq 0 to 31 Clear Enable Register. Use the Interrupt Clear-Enable Registers to disable interrupts and determine which interrupts are currently enabled.

**Figure 2-22. ICER0 Register**

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
CLRENA																															
R/W-0h																															

**Table 2-28. ICER0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLRENA	R/W	0h	Writing 0 to a CLRENA bit has no effect, writing 1 to a bit disables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the CLRENA field.

### 2.4.3.4 ICER1 Register (Offset = 184h) [reset = 00000000h]

ICER1 is shown in [Figure 2-23](#) and described in [Table 2-29](#).

Irq 32 to 63 Clear Enable Register. Use the Interrupt Clear-Enable Registers to disable interrupts and determine which interrupts are currently enabled.

**Figure 2-23. ICER1 Register**

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
CLRENA																															
R/W-0h																															

**Table 2-29. ICER1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLRENA	R/W	0h	Writing 0 to a CLRENA bit has no effect, writing 1 to a bit disables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the CLRENA field.

#### 2.4.3.5 ISPR0 Register (Offset = 200h) [reset = 00000000h]

ISPR0 is shown in [Figure 2-24](#) and described in [Table 2-30](#).

Irq 0 to 31 Set Pending Register. Use the Interrupt Set-Pending Registers to force interrupts into the pending state and determine which interrupts are currently pending

**Figure 2-24. ISPR0 Register**

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
SETPEND																															
R/W-0h																															

**Table 2-30. ISPR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SETPEND	R/W	0h	Writing 0 to a SETPEND bit has no effect, writing 1 to a bit pends the corresponding interrupt. Reading the bit returns its current state.

#### 2.4.3.6 ISPR1 Register (Offset = 204h) [reset = 00000000h]

ISPR1 is shown in [Figure 2-25](#) and described in [Table 2-31](#).

Irq 32 to 63 Set Pending Register. Use the Interrupt Set-Pending Registers to force interrupts into the pending state and determine which interrupts are currently pending

**Figure 2-25. ISPR1 Register**

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
SETPEND																															
R/W-0h																															

**Table 2-31. ISPR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SETPEND	R/W	0h	Writing 0 to a SETPEND bit has no effect, writing 1 to a bit pends the corresponding interrupt. Reading the bit returns its current state.

#### 2.4.3.7 ICPR0 Register (Offset = 280h) [reset = 00000000h]

ICPR0 is shown in Figure 2-26 and described in Table 2-32.

Irq 0 to 31 Clear Pending Register. Use the Interrupt Clear-Pending Registers to clear pending interrupts and determine which interrupts are currently pending.

**Figure 2-26. ICPR0 Register**

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
CLRPEND																															
R/W-0h																															

**Table 2-32. ICPR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLRPEND	R/W	0h	Writing 0 to a CLRPEND bit has no effect, writing 1 to a bit clears the corresponding pending interrupt. Reading the bit returns its current state.

#### 2.4.3.8 ICPR1 Register (Offset = 284h) [reset = 00000000h]

ICPR1 is shown in Figure 2-27 and described in Table 2-33.

Irq 32 to 63 Clear Pending Register. Use the Interrupt Clear-Pending Registers to clear pending interrupts and determine which interrupts are currently pending.

**Figure 2-27. ICPR1 Register**

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
CLRPEND																															
R/W-0h																															

**Table 2-33. ICPR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLRPEND	R/W	0h	Writing 0 to a CLRPEND bit has no effect, writing 1 to a bit clears the corresponding pending interrupt. Reading the bit returns its current state.

### 2.4.3.9 IABR0 Register (Offset = 300h) [reset = 00000000h]

IABR0 is shown in [Figure 2-28](#) and described in [Table 2-34](#).

Irq 0 to 31 Active Bit Register. Read the Active Bit Registers to determine which interrupts are active. Each flag in the register corresponds to one interrupt.

**Figure 2-28. IABR0 Register**

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
ACTIVE																															
R-0h																															

**Table 2-34. IABR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ACTIVE	R	0h	Interrupt active flags. Reading 0 implies the interrupt is not active or stacked. Reading 1 implies the interrupt is active or pre-empted and stacked.

### 2.4.3.10 IABR1 Register (Offset = 304h) [reset = 00000000h]

IABR1 is shown in [Figure 2-29](#) and described in [Table 2-35](#).

Irq 32 to 63 Active Bit Register. Read the Active Bit Registers to determine which interrupts are active. Each flag in the register corresponds to one interrupt.

**Figure 2-29. IABR1 Register**

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
ACTIVE																															
R-0h																															

**Table 2-35. IABR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ACTIVE	R	0h	Interrupt active flags. Reading 0 implies the interrupt is not active or stacked. Reading 1 implies the interrupt is active or pre-empted and stacked.

#### 2.4.3.11 IPR0 Register (Offset = 400h) [reset = 00000000h]

IPR0 is shown in [Figure 2-30](#) and described in [Table 2-36](#).

Irq 0 to 3 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-30. IPR0 Register**

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
PRI_3								PRI_2								PRI_1								PRI_0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 2-36. IPR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_3	R/W	0h	Priority of interrupt 3
23-16	PRI_2	R/W	0h	Priority of interrupt 2
15-8	PRI_1	R/W	0h	Priority of interrupt 1
7-0	PRI_0	R/W	0h	Priority of interrupt 0

#### 2.4.3.12 IPR1 Register (Offset = 404h) [reset = 00000000h]

IPR1 is shown in [Figure 2-31](#) and described in [Table 2-37](#).

Irq 4 to 7 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-31. IPR1 Register**

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
PRI_7								PRI_6								PRI_5								PRI_4							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 2-37. IPR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_7	R/W	0h	Priority of interrupt 7
23-16	PRI_6	R/W	0h	Priority of interrupt 6
15-8	PRI_5	R/W	0h	Priority of interrupt 5
7-0	PRI_4	R/W	0h	Priority of interrupt 4

#### 2.4.3.13 IPR2 Register (Offset = 408h) [reset = 00000000h]

IPR2 is shown in [Figure 2-32](#) and described in [Table 2-38](#).

Irq 8 to 11 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-32. IPR2 Register**

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
PRI_11						PRI_10						PRI_9						PRI_8													
R/W-0h						R/W-0h						R/W-0h						R/W-0h													

**Table 2-38. IPR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_11	R/W	0h	Priority of interrupt 11
23-16	PRI_10	R/W	0h	Priority of interrupt 10
15-8	PRI_9	R/W	0h	Priority of interrupt 9
7-0	PRI_8	R/W	0h	Priority of interrupt 8

#### 2.4.3.14 IPR3 Register (Offset = 40Ch) [reset = 00000000h]

IPR3 is shown in [Figure 2-33](#) and described in [Table 2-39](#).

Irq 12 to 15 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-33. IPR3 Register**

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
PRI_15						PRI_14						PRI_13						PRI_12													
R/W-0h						R/W-0h						R/W-0h						R/W-0h													

**Table 2-39. IPR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_15	R/W	0h	Priority of interrupt 15
23-16	PRI_14	R/W	0h	Priority of interrupt 14
15-8	PRI_13	R/W	0h	Priority of interrupt 13
7-0	PRI_12	R/W	0h	Priority of interrupt 12

#### 2.4.3.15 IPR4 Register (Offset = 410h) [reset = 00000000h]

IPR4 is shown in [Figure 2-34](#) and described in [Table 2-40](#).

Irq 16 to 19 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-34. IPR4 Register**

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
PRI_19							PRI_18							PRI_17							PRI_16										
R/W-0h							R/W-0h							R/W-0h							R/W-0h										

**Table 2-40. IPR4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_19	R/W	0h	Priority of interrupt 19
23-16	PRI_18	R/W	0h	Priority of interrupt 18
15-8	PRI_17	R/W	0h	Priority of interrupt 17
7-0	PRI_16	R/W	0h	Priority of interrupt 16

#### 2.4.3.16 IPR5 Register (Offset = 414h) [reset = 00000000h]

IPR5 is shown in [Figure 2-35](#) and described in [Table 2-41](#).

Irq 20 to 23 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-35. IPR5 Register**

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
PRI_23							PRI_22							PRI_21							PRI_20										
R/W-0h							R/W-0h							R/W-0h							R/W-0h										

**Table 2-41. IPR5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_23	R/W	0h	Priority of interrupt 23
23-16	PRI_22	R/W	0h	Priority of interrupt 22
15-8	PRI_21	R/W	0h	Priority of interrupt 21
7-0	PRI_20	R/W	0h	Priority of interrupt 20

#### 2.4.3.17 IPR6 Register (Offset = 418h) [reset = 00000000h]

IPR6 is shown in [Figure 2-36](#) and described in [Table 2-42](#).

Irq 24 to 27 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-36. IPR6 Register**

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
PRI_27								PRI_26								PRI_25								PRI_24							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 2-42. IPR6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_27	R/W	0h	Priority of interrupt 27
23-16	PRI_26	R/W	0h	Priority of interrupt 26
15-8	PRI_25	R/W	0h	Priority of interrupt 25
7-0	PRI_24	R/W	0h	Priority of interrupt 24

#### 2.4.3.18 IPR7 Register (Offset = 41Ch) [reset = 00000000h]

IPR7 is shown in [Figure 2-37](#) and described in [Table 2-43](#).

Irq 28 to 31 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-37. IPR7 Register**

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
PRI_31								PRI_30								PRI_29								PRI_28							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 2-43. IPR7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_31	R/W	0h	Priority of interrupt 31
23-16	PRI_30	R/W	0h	Priority of interrupt 30
15-8	PRI_29	R/W	0h	Priority of interrupt 29
7-0	PRI_28	R/W	0h	Priority of interrupt 28

#### 2.4.3.19 IPR8 Register (Offset = 420h) [reset = 00000000h]

IPR8 is shown in [Figure 2-38](#) and described in [Table 2-44](#).

Irq 32 to 35 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-38. IPR8 Register**

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
PRI_35								PRI_34								PRI_33								PRI_32							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 2-44. IPR8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_35	R/W	0h	Priority of interrupt 35
23-16	PRI_34	R/W	0h	Priority of interrupt 34
15-8	PRI_33	R/W	0h	Priority of interrupt 33
7-0	PRI_32	R/W	0h	Priority of interrupt 32

#### 2.4.3.20 IPR9 Register (Offset = 424h) [reset = 00000000h]

IPR9 is shown in [Figure 2-39](#) and described in [Table 2-45](#).

Irq 36 to 39 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-39. IPR9 Register**

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
PRI_39								PRI_38								PRI_37								PRI_36							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 2-45. IPR9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_39	R/W	0h	Priority of interrupt 39
23-16	PRI_38	R/W	0h	Priority of interrupt 38
15-8	PRI_37	R/W	0h	Priority of interrupt 37
7-0	PRI_36	R/W	0h	Priority of interrupt 36

#### 2.4.3.21 IPR10 Register (Offset = 428h) [reset = 00000000h]

IPR10 is shown in [Figure 2-40](#) and described in [Table 2-46](#).

Irq 40 to 43 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-40. IPR10 Register**

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
PRI_43							PRI_42							PRI_41							PRI_40										
R/W-0h							R/W-0h							R/W-0h							R/W-0h										

**Table 2-46. IPR10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_43	R/W	0h	Priority of interrupt 43
23-16	PRI_42	R/W	0h	Priority of interrupt 42
15-8	PRI_41	R/W	0h	Priority of interrupt 41
7-0	PRI_40	R/W	0h	Priority of interrupt 40

#### 2.4.3.22 IPR11 Register (Offset = 42Ch) [reset = 00000000h]

IPR11 is shown in [Figure 2-41](#) and described in [Table 2-47](#).

Irq 44 to 47 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-41. IPR11 Register**

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
PRI_47							PRI_46							PRI_45							PRI_44										
R/W-0h							R/W-0h							R/W-0h							R/W-0h										

**Table 2-47. IPR11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_47	R/W	0h	Priority of interrupt 47
23-16	PRI_46	R/W	0h	Priority of interrupt 46
15-8	PRI_45	R/W	0h	Priority of interrupt 45
7-0	PRI_44	R/W	0h	Priority of interrupt 44

#### 2.4.3.23 IPR12 Register (Offset = 430h) [reset = 00000000h]

IPR12 is shown in [Figure 2-42](#) and described in [Table 2-48](#).

Irq 48 to 51 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-42. IPR12 Register**

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
PRI_51							PRI_50							PRI_49							PRI_48										
R/W-0h							R/W-0h							R/W-0h							R/W-0h										

**Table 2-48. IPR12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_51	R/W	0h	Priority of interrupt 51
23-16	PRI_50	R/W	0h	Priority of interrupt 50
15-8	PRI_49	R/W	0h	Priority of interrupt 49
7-0	PRI_48	R/W	0h	Priority of interrupt 48

#### 2.4.3.24 IPR13 Register (Offset = 434h) [reset = 00000000h]

IPR13 is shown in [Figure 2-43](#) and described in [Table 2-49](#).

Irq 52 to 55 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-43. IPR13 Register**

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
PRI_55							PRI_54							PRI_53							PRI_52										
R/W-0h							R/W-0h							R/W-0h							R/W-0h										

**Table 2-49. IPR13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_55	R/W	0h	Priority of interrupt 55
23-16	PRI_54	R/W	0h	Priority of interrupt 54
15-8	PRI_53	R/W	0h	Priority of interrupt 53
7-0	PRI_52	R/W	0h	Priority of interrupt 52

#### 2.4.3.25 IPR14 Register (Offset = 438h) [reset = 00000000h]

IPR14 is shown in [Figure 2-44](#) and described in [Table 2-50](#).

Irq 56 to 59 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-44. IPR14 Register**

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
PRI_59							PRI_58							PRI_57							PRI_56										
R/W-0h							R/W-0h							R/W-0h							R/W-0h										

**Table 2-50. IPR14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_59	R/W	0h	Priority of interrupt 59
23-16	PRI_58	R/W	0h	Priority of interrupt 58
15-8	PRI_57	R/W	0h	Priority of interrupt 57
7-0	PRI_56	R/W	0h	Priority of interrupt 56

#### 2.4.3.26 IPR15 Register (Offset = 43Ch) [reset = 00000000h]

IPR15 is shown in [Figure 2-45](#) and described in [Table 2-51](#).

Irq 60 to 63 Priority Register. Use the Interrupt Priority Registers to assign a priority from 0 to 255 to each of the available interrupts. 0 is the highest priority, and 255 is the lowest.

The hardware priority mechanism only looks at the upper N bits of the priority field (where N is 3 for the MSP432 family), so any prioritization must be performed in those bits. The remaining bits can be used to sub-prioritize the interrupt sources, and may be used by the hardware priority mechanism on a future part

**Figure 2-45. IPR15 Register**

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
PRI_63							PRI_62							PRI_61							PRI_60										
R/W-0h							R/W-0h							R/W-0h							R/W-0h										

**Table 2-51. IPR15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_63	R/W	0h	Priority of interrupt 63
23-16	PRI_62	R/W	0h	Priority of interrupt 62
15-8	PRI_61	R/W	0h	Priority of interrupt 61
7-0	PRI_60	R/W	0h	Priority of interrupt 60

#### 2.4.3.27 STIR Register (Offset = F00h) [reset = 00000000h]

STIR is shown in [Figure 2-46](#) and described in [Table 2-52](#).

Software Trigger Interrupt Register. Use the Software Trigger Interrupt Register to pend an interrupt to trigger.

**Figure 2-46. STIR Register**

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
RESERVED																								INTID							
W-0h																								W-0h							

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

Bit	Field	Type	Reset	Description
31-9	RESERVED	W	0h	
8-0	INTID	W	0h	Interrupt ID field. Writing a value to the INTID field is the same as manually pending an interrupt by setting the corresponding interrupt bit in an Interrupt Set Pending Register.

## 2.4.4 SYSTICK Registers

Table 2-53 lists the memory-mapped registers for the SYSTICK. All register offset addresses not listed in Table 2-53 should be considered as reserved locations and the register contents should not be modified.

**Table 2-53. SYSTICK Registers**

Offset	Acronym	Register Name	Type	Reset	Section
10h	STCSR	SysTick Control and Status Register	read-write	00000004h	Section 2.4.4.1
14h	STRVR	SysTick Reload Value Register	read-write	Undefined	Section 2.4.4.2
18h	STCVR	SysTick Current Value Register	read-write	Undefined	Section 2.4.4.3
1Ch	STCR	SysTick Calibration Value Register	read-only	Undefined	Section 2.4.4.4

#### 2.4.4.1 STCSR Register (Offset = 10h) [reset = 00000004h]

STCSR is shown in [Figure 2-47](#) and described in [Table 2-54](#).

SysTick Control and Status Register. Use the SysTick Control and Status Register to enable the SysTick features.

**Figure 2-47. STCSR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED						COUNTFLAG	
R/W-0h						R-0h	
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED					CLKSOURCE	TICKINT	ENABLE
R/W-0h					R-1h	R/W-0h	R/W-0h

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

Bit	Field	Type	Reset	Description
31-17	RESERVED	R/W	0h	
16	COUNTFLAG	R	0h	Returns 1 if timer counted to 0 since last time this was read. Clears on read by application of any part of the SysTick Control and Status Register. If read by the debugger using the DAP, this bit is cleared on read-only if the MasterType bit in the AHB-AP Control Register is set to 0. Otherwise, the COUNTFLAG bit is not changed by the debugger read.
15-3	RESERVED	R/W	0h	
2	CLKSOURCE	R	1h	Clock source. 0b = Not applicable 1b = Core clock
1	TICKINT	R/W	0h	
0	ENABLE	R/W	0h	Enable SysTick counter 0b (R/W) = Counter disabled

#### 2.4.4.2 STRVR Register (Offset = 14h)

STRVR is shown in [Figure 2-48](#) and described in [Table 2-55](#).

SysTick Reload Value Register. Use the SysTick Reload Value Register to specify the start value to load into the current value register when the counter reaches 0. It can be any value between 1 and 0x00FFFFFF. A start value of 0 is possible, but has no effect because the SysTick interrupt and COUNTFLAG are activated when counting from 1 to 0.

**Figure 2-48. STRVR Register**

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
RESERVED																											RELOAD				
R/W																											R/W				

**Table 2-55. STRVR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R/W	Undefined	
23-0	RELOAD	R/W	Undefined	Value to load into the SysTick Current Value Register when the counter reaches 0.

#### 2.4.4.3 STCVR Register (Offset = 18h)

STCVR is shown in [Figure 2-49](#) and described in [Table 2-56](#).

SysTick Current Value Register. Use the SysTick Current Value Register to find the current value in the register.

**Figure 2-49. STCVR Register**

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	
RESERVED																																
R/W																																

**Table 2-56. STCVR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R/W	Undefined	
23-0	CURRENT	R/W	Undefined	Current value at the time the register is accessed. No read-modify-write protection is provided, so change with care. Writing to it with any value clears the register to 0. Clearing this register also clears the COUNTFLAG bit of the SysTick Control and Status Register.

#### 2.4.4.4 STCR Register (Offset = 1Ch)

STCR is shown in [Figure 2-50](#) and described in [Table 2-57](#).

SysTick Calibration Value Register. Use the SysTick Calibration Value Register to enable software to scale to any required speed using divide and multiply.

**Figure 2-50. STCR Register**

31	30	29	28	27	26	25	24
NOREF	SKEW			RESERVED			
R-	R-			R-			
23	22	21	20	19	18	17	16
			TENMS				
			R-				
15	14	13	12	11	10	9	8
			TENMS				
			R-				
7	6	5	4	3	2	1	0
			TENMS				
			R-				

**Table 2-57. STCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	NOREF	R	Undefined	Reads as one. Indicates that no separate reference clock is provided.
30	SKEW	R	Undefined	Reads as one. The calibration value is not exactly 10ms because of clock frequency. This could affect its suitability as a software real time clock.
29-24	RESERVED	R	Undefined	
23-0	TENMS	R	Undefined	Reads as zero. Indicates calibration value is not known.

## 2.4.5 SCB Registers

Table 2-58 lists the memory-mapped registers for the SCB. All register offset addresses not listed in Table 2-58 should be considered as reserved locations and the register contents should not be modified.

**Table 2-58. SCB Registers**

Offset	Acronym	Register Name	Type	Reset	Section
D00h	CPUID	CPUID Base Register	read-only	410FC241h	<a href="#">Section 2.4.5.1</a>
D04h	ICSR	Interrupt Control State Register	read-write	00000000h	<a href="#">Section 2.4.5.2</a>
D08h	VTOR	Vector Table Offset Register	read-write	00000000h	<a href="#">Section 2.4.5.3</a>
D0Ch	AIRCR	Application Interrupt/Reset Control Register	read-write	FA050000h	<a href="#">Section 2.4.5.4</a>
D10h	SCR	System Control Register	read-write	00000000h	<a href="#">Section 2.4.5.5</a>
D14h	CCR	Configuration Control Register	read-write	00000200h	<a href="#">Section 2.4.5.6</a>
D18h	SHPR1	System Handlers 4-7 Priority Register	read-write	00000000h	<a href="#">Section 2.4.5.7</a>
D1Ch	SHPR2	System Handlers 8-11 Priority Register	read-write	00000000h	<a href="#">Section 2.4.5.8</a>
D20h	SHPR3	System Handlers 12-15 Priority Register	read-write	00000000h	<a href="#">Section 2.4.5.9</a>
D24h	SHCSR	System Handler Control and State Register	read-write	00000000h	<a href="#">Section 2.4.5.10</a>
D28h	CFSR	Configurable Fault Status Registers	read-write	00000000h	<a href="#">Section 2.4.5.11</a>
D2Ch	HFSR	Hard Fault Status Register	read-write	00000000h	<a href="#">Section 2.4.5.12</a>
D30h	DFSR	Debug Fault Status Register	read-write	00000000h	<a href="#">Section 2.4.5.13</a>
D34h	MMFAR	Mem Manage Fault Address Register	read-write	Undefined	<a href="#">Section 2.4.5.14</a>
D38h	BFAR	Bus Fault Address Register	read-write	Undefined	<a href="#">Section 2.4.5.15</a>
D3Ch	AFSR	Auxiliary Fault Status Register	read-write	00000000h	<a href="#">Section 2.4.5.16</a>
D40h	PFR0	Processor Feature register0	read-only	00000030h	<a href="#">Section 2.4.5.17</a>
D44h	PFR1	Processor Feature register1	read-only	00000200h	<a href="#">Section 2.4.5.18</a>
D48h	DFR0	Debug Feature register0	read-only	00100000h	<a href="#">Section 2.4.5.19</a>
D4Ch	AFR0	Auxiliary Feature register0	read-only	00000000h	<a href="#">Section 2.4.5.20</a>
D50h	MMFR0	Memory Model Feature register0	read-only	00100030h	<a href="#">Section 2.4.5.21</a>
D54h	MMFR1	Memory Model Feature register1	read-only	00000000h	<a href="#">Section 2.4.5.22</a>
D58h	MMFR2	Memory Model Feature register2	read-only	01000000h	<a href="#">Section 2.4.5.23</a>
D5Ch	MMFR3	Memory Model Feature register3	read-only	00000000h	<a href="#">Section 2.4.5.24</a>
D60h	ISAR0	ISA Feature register0	read-only	01101110h	<a href="#">Section 2.4.5.25</a>
D64h	ISAR1	ISA Feature register1	read-only	02112000h	<a href="#">Section 2.4.5.26</a>
D68h	ISAR2	ISA Feature register2	read-only	21232231h	<a href="#">Section 2.4.5.27</a>
D6Ch	ISAR3	ISA Feature register3	read-only	01111131h	<a href="#">Section 2.4.5.28</a>
D70h	ISAR4	ISA Feature register4	read-only	01310132h	<a href="#">Section 2.4.5.29</a>
D88h	CPACR	Coprocessor Access Control Register	read-write	00F00000h	<a href="#">Section 2.4.5.30</a>

### 2.4.5.1 CPUID Register (Offset = D00h) [reset = 410FC241h]

CPUID is shown in [Figure 2-51](#) and described in [Table 2-59](#).

CPUID Base Register. Read the CPU ID Base Register to determine: the ID number of the processor core, the version number of the processor core, the implementation details of the processor core.

**Figure 2-51. CPUID Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IMPLEMENTER						VARIANT				CONSTANT					
R-41h										R-0h				R-Fh	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PARTNO						REVISION						R-1h			
R-C24h															

**Table 2-59. CPUID Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	IMPLEMENTER	R	41h	Implementor code.
23-20	VARIANT	R	0h	Implementation defined variant number.
19-16	CONSTANT	R	Fh	Reads as 0xC
15-4	PARTNO	R	C24h	Number of processor within family.
3-0	REVISION	R	1h	Implementation defined revision number.

#### 2.4.5.2 ICSR Register (Offset = D04h) [reset = 00000000h]

ICSR is shown in [Figure 2-52](#) and described in [Table 2-60](#).

Interrupt Control State Register. Use the Interrupt Control State Register to set a pending Non-Maskable Interrupt (NMI), set or clear a pending SVC, set or clear a pending SysTick, check for pending exceptions, check the vector number of the highest priority pending exception, check the vector number of the active exception.

**Figure 2-52. ICSR Register**

31	30	29	28	27	26	25	24
NMIPENDSET	RESERVED	PENDSVSET	PENDSVCLR	PENDSTSET	PENDSTCLR	RESERVED	
R/W-0h	R/W-0h	R/W-0h	W-0h	R/W-0h	W-0h	R/W-0h	
23	22	21	20	19	18	17	16
ISRPREEMPT	ISRPENDING	RESERVED			VECTPENDING		
R-0h	R-0h	R/W-0h			R-0h		
15	14	13	12	11	10	9	8
VECTPENDING		RETTTOBASE		RESERVED		VECTACTIVE	
	R-0h	R-0h	R/W-0h		R-0h		
7	6	5	4	3	2	1	0
VECTACTIVE							
	R-0h						

**Table 2-60. ICSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	NMIPENDSET	R/W	0h	Set pending NMI bit. NMIPENDSET pends and activates an NMI. Because NMI is the highest-priority interrupt, it takes effect as soon as it registers. 0b (R/W) = do not set pending NMI 1b (R/W) = set pending NMI
30-29	RESERVED	R/W	0h	
28	PENDSVSET	R/W	0h	Set pending pendSV bit. 0b (R/W) = do not set pending pendSV 1b (R/W) = set pending PendSV
27	PENDSVCLR	W	0h	Clear pending pendSV bit 0b (R/W) = do not clear pending pendSV 1b (R/W) = clear pending pendSV
26	PENDSTSET	R/W	0h	Set a pending SysTick bit. 0b (R/W) = do not set pending SysTick 1b (R/W) = set pending SysTick
25	PENDSTCLR	W	0h	Clear pending SysTick bit 0b (R/W) = do not clear pending SysTick 1b (R/W) = clear pending SysTick
24	RESERVED	R/W	0h	
23	ISRPREEMPT	R	0h	You must only use this at debug time. It indicates that a pending interrupt is to be taken in the next running cycle. If C_MASKINTS is clear in the Debug Halting Control and Status Register, the interrupt is serviced. 0b (R/W) = a pending exception is not serviced. 1b (R/W) = a pending exception is serviced on exit from the debug halt state
22	ISRPENDING	R	0h	Interrupt pending flag. Excludes NMI and faults. 0b (R/W) = interrupt not pending 1b (R/W) = interrupt pending
21-18	RESERVED	R/W	0h	

**Table 2-60. ICSR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
17-12	VECTPENDING	R	0h	Pending ISR number field. VECTPENDING contains the interrupt number of the highest priority pending ISR.
11	RETTTOBASE	R	0h	This bit is 1 when the set of all active exceptions minus the IPSR_current_exception yields the empty set.
10-9	RESERVED	R/W	0h	
8-0	VECTACTIVE	R	0h	Active ISR number field. Reset clears the VECTACTIVE field.

### 2.4.5.3 VTOR Register (Offset = D08h) [reset = 00000000h]

VTOR is shown in [Figure 2-53](#) and described in [Table 2-61](#).

Vector Table Offset Register. Use the Vector Table Offset Register to determine: if the vector table is in RAM or code memory, the vector table offset.

**Figure 2-53. VTOR Register**

31	30	29	28	27	26	25	24
RESERVED	TBLBASE			TBLOFF			
R/W-0h	R/W-0h			R/W-0h			
23	22	21	20	19	18	17	16
			TBLOFF				
			R/W-0h				
15	14	13	12	11	10	9	8
			TBLOFF				
			R/W-0h				
7	6	5	4	3	2	1	0
TBLOFF			RESERVED				
R/W-0h			R/W-0h				

**Table 2-61. VTOR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	
29	TBLBASE	R/W	0h	Table base is in Code (0) or RAM (1).
28-7	TBLOFF	R/W	0h	Vector table base offset field. Contains the offset of the table base from the bottom of the SRAM or CODE space.
6-0	RESERVED	R/W	0h	

#### 2.4.5.4 AIRCR Register (Offset = D0Ch) [reset = FA050000h]

Register mask: FFFF7FFFh

AIRCR is shown in [Figure 2-54](#) and described in [Table 2-62](#).

Application Interrupt/Reset Control Register. Use the Application Interrupt and Reset Control Register to: determine data endianness, clear all active state information for debug or to recover from a hard failure, execute a system reset, alter the priority grouping position (binary point).

**Figure 2-54. AIRCR Register**

31	30	29	28	27	26	25	24
			VECTKEY				
			W-FA05h				
23	22	21	20	19	18	17	16
		VECTKEY					
			W-FA05h				
15	14	13	12	11	10	9	8
ENDIANESS		RESERVED			PRIGROUP		
R-0		R/W-0h			R/W-0h		
7	6	5	4	3	2	1	0
		RESERVED		SYSRESETREQ	VECTCLRACTIVE	VECTRESET	
		R/W-0h		W-0h	W-0h	W-0h	

**Table 2-62. AIRCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-16	VECTKEY	W	FA05h	Register key. Writing to this register requires 0x5FA in the VECTKEY field. Otherwise the write value is ignored.
15	ENDIANESS	R	0h	Data endianness bit. ENDIANESS is sampled from the BIGEND input port during reset. You cannot change ENDIANESS outside of reset. 0b (R/W) = little endian 1b (R/W) = big endian
14-11	RESERVED	R/W	0h	
10-8	PRIGROUP	R/W	0h	Interrupt priority grouping field. The PRIGROUP field is a binary point position indicator for creating subpriorities for exceptions that share the same preemption level. It divides the PRI_n field in the Interrupt Priority Register into a preemption level and a subpriority level. The binary point is a left-of value. This means that the PRIGROUP value represents a point starting at the left of the Least Significant Bit (LSB). This is bit [0] of 7:0. The lowest value might not be 0 depending on the number of bits allocated for priorities, and implementation choices
7-3	RESERVED	R/W	0h	
2	SYSRESETREQ	W	0h	Causes a signal to be asserted to the outer system that indicates a reset is requested. Intended to force a large system reset of all major components except for debug. Setting this bit does not prevent Halting Debug from running.
1	VECTCLRACTIVE	W	0h	Clears all active state information for active NMI, fault, and interrupts. It is the responsibility of the application to reinitialize the stack. The VECTCLRACTIVE bit is for returning to a known state during debug. The VECTCLRACTIVE bit self-clears. IPSR is not cleared by this operation. So, if used by an application, it must only be used at the base level of activation, or within a system handler whose active bit can be set.

**Table 2-62. AIRCR Register Field Descriptions (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
0	VECTRESET	W	0h	System Reset bit. Resets the system, with the exception of debug components. The VECTRESET bit self-clears. Reset clears the VECTRESET bit. For debugging, only write this bit when the core is halted.

#### 2.4.5.5 SCR Register (Offset = D10h) [reset = 00000000h]

SCR is shown in [Figure 2-55](#) and described in [Table 2-63](#).

System Control Register. Use the System Control Register for power-management functions: signal to the system when the processor can enter a low power state, control how the processor enters and exits low power states.

**Figure 2-55. SCR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		SEVONPEND		RESERVED		SLEEPDEEP	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 2-63. SCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	0h	
4	SEVONPEND	R/W	0h	When enabled, this causes WFE to wake up when an interrupt moves from inactive to pended. Otherwise, WFE only wakes up from an event signal, external and SEV instruction generated. The event input, RXEV, is registered even when not waiting for an event, and so effects the next WFE.
3	RESERVED	R/W	0h	
2	SLEEPDEEP	R/W	0h	Sleep deep bit. 0b (R/W) = not OK to turn off system clock 1b (R/W) = indicates to the system that Cortex-M4 clock can be stopped. Setting this bit causes the SLEEPDEEP port to be asserted when the processor can be stopped.
1	SLEEPONEXIT	R/W	0h	Sleep on exit when returning from Handler mode to Thread mode. Enables interrupt driven applications to avoid returning to empty main application. 0b (R/W) = do not sleep when returning to thread mode 1b (R/W) = sleep on ISR exit
0	RESERVED	R/W	0h	

#### 2.4.5.6 CCR Register (Offset = D14h) [reset = 00000200h]

CCR is shown in [Figure 2-56](#) and described in [Table 2-64](#).

Configuration Control Register. Use the Configuration Control Register to: enable NMI, HardFault and FAULTMASK to ignore bus fault, trap divide by zero and unaligned accesses, enable user access to the Software Trigger Exception Register, control entry to Thread Mode.

**Figure 2-56. CCR Register**

31	30	29	28	27	26	25	24	
RESERVED								
R/W-0h								
23	22	21	20	19	18	17	16	
RESERVED								
R/W-0h								
15	14	13	12	11	10	9	8	
RESERVED						STKALIGN	BFHFNIGN	
R/W-0h						R/W-1h	R/W-0h	
7	6	5	4	3	2	1	0	
RESERVED			DIV_0_TRP	UNALIGN_TRP	RESERVED	USERSETMPEND	NONBASETHREDENA	
R/W-0h			R/W-0h		R/W-0h		R/W-0h	
R/W-0h			R/W-0h		R/W-0h		R/W-0h	

**Table 2-64. CCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R/W	0h	
9	STKALIGN	R/W	1h	Stack alignment bit. 0b (R/W) = Only 4-byte alignment is ensured for the SP used prior to the exception on exception entry. 1b (R/W) = On exception entry, the SP used prior to the exception is adjusted to be 8-byte aligned and the context to restore it is saved. The SP is restored on the associated exception return.
8	BFHFNIGN	R/W	0h	When enabled, this causes handlers running at priority -1 and -2 (Hard Fault, NMI, and FAULTMASK escalated handlers) to ignore Data Bus faults caused by load and store instructions. When disabled, these bus faults cause a lock-up. You must only use this enable with extreme caution. All data bus faults are ignored therefore you must only use it when the handler and its data are in absolutely safe memory. Its normal use is to probe system devices and bridges to detect control path problems and fix them.
7-5	RESERVED	R/W	0h	
4	DIV_0_TRP	R/W	0h	Trap on Divide by 0. This enables faulting/halting when an attempt is made to divide by 0. The relevant Usage Fault Status Register bit is DIVBYZERO.
3	UNALIGN_TRP	R/W	0h	Trap for unaligned access. This enables faulting/halting on any unaligned half or full word access. Unaligned load-store multiples always fault. The relevant Usage Fault Status Register bit is UNALIGNED.
2	RESERVED	R/W	0h	
1	USERSETMPEND	R/W	0h	If written as 1, enables user code to write the Software Trigger Interrupt register to trigger (pend) a Main exception, which is one associated with the Main stack pointer.
0	NONBASETHREDENA	R/W	0h	When 0, default, It is only possible to enter Thread mode when returning from the last exception. When set to 1, Thread mode can be entered from any level in Handler mode by controlled return value.

#### 2.4.5.7 SHPR1 Register (Offset = D18h) [reset = 00000000h]

SHPR1 is shown in [Figure 2-57](#) and described in [Table 2-65](#).

System Handlers 4-7 Priority Register. Use the three System Handler Priority Registers to prioritize the following system handlers: memory manage, bus fault, usage fault, debug monitor, SVC, SysTick, PendSV. System Handlers are a special class of exception handler that can have their priority set to any of the priority levels. Most can be masked on (enabled) or off (disabled). When disabled, the fault is always treated as a Hard Fault.

**Figure 2-57. SHPR1 Register**

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
PRI_7								PRI_6								PRI_5								PRI_4							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 2-65. SHPR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_7	R/W	0h	Priority of system handler 7.
23-16	PRI_6	R/W	0h	Priority of system handler 6.
15-8	PRI_5	R/W	0h	Priority of system handler 5.
7-0	PRI_4	R/W	0h	Priority of system handler 4.

#### 2.4.5.8 SHPR2 Register (Offset = D1Ch) [reset = 00000000h]

SHPR2 is shown in [Figure 2-58](#) and described in [Table 2-66](#).

System Handlers 8-11 Priority Register. Use the three System Handler Priority Registers to prioritize the following system handlers: memory manage, bus fault, usage fault, debug monitor, SVC, SysTick, PendSV. System Handlers are a special class of exception handler that can have their priority set to any of the priority levels. Most can be masked on (enabled) or off (disabled). When disabled, the fault is always treated as a Hard Fault.

**Figure 2-58. SHPR2 Register**

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
PRI_11								PRI_10								PRI_9								PRI_8							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 2-66. SHPR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_11	R/W	0h	Priority of system handler 11.
23-16	PRI_10	R/W	0h	Priority of system handler 10.
15-8	PRI_9	R/W	0h	Priority of system handler 9.
7-0	PRI_8	R/W	0h	Priority of system handler 8.

#### 2.4.5.9 SHPR3 Register (Offset = D20h) [reset = 00000000h]

SHPR3 is shown in [Figure 2-59](#) and described in [Table 2-67](#).

System Handlers 12-15 Priority Register. Use the three System Handler Priority Registers to prioritize the following system handlers: memory manage, bus fault, usage fault, debug monitor, SVC, SysTick, PendSV. System Handlers are a special class of exception handler that can have their priority set to any of the priority levels. Most can be masked on (enabled) or off (disabled). When disabled, the fault is always treated as a Hard Fault.

**Figure 2-59. SHPR3 Register**

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
PRI_15								PRI_14								PRI_13								PRI_12							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

**Table 2-67. SHPR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	PRI_15	R/W	0h	Priority of system handler 15.
23-16	PRI_14	R/W	0h	Priority of system handler 14.
15-8	PRI_13	R/W	0h	Priority of system handler 13.
7-0	PRI_12	R/W	0h	Priority of system handler 12.

#### 2.4.5.10 SHCSR Register (Offset = D24h) [reset = 00000000h]

SHCSR is shown in [Figure 2-60](#) and described in [Table 2-68](#).

System Handler Control and State Register. Use the System Handler Control and State Register to: enable or disable the system handlers, determine the pending status of bus fault, mem manage fault, and SVC, determine the active status of the system handlers. If a fault condition occurs while its fault handler is disabled, the fault escalates to a Hard Fault.

**Figure 2-60. SHCSR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED					USGFAULTENA	BUSFAULTENA	MEMFAULTENA
R/W-0h					R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
SVCALLPENDED	BUSFAULTPEN DED	MEMFAULTPEN DED	USGFAULTPEN DED	SYSTICKACT	PENDSVACT	RESERVED	MONITORACT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R/W-0h	R-0h
7	6	5	4	3	2	1	0
SVCALLACT	RESERVED			USGFAULTACT	RESERVED	BUSFAULTACT	MEMFAULTACT
R-0h	R/W-0h			R-0h	R/W-0h	R-0h	R-0h

**Table 2-68. SHCSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-19	RESERVED	R/W	0h	
18	USGFAULTENA	R/W	0h	Usage fault system handler enable 0b (R/W) = disabled 1b (R/W) = enabled
17	BUSFAULTENA	R/W	0h	Bus fault system handler enable 0b (R/W) = disabled 1b (R/W) = enabled
16	MEMFAULTENA	R/W	0h	MemManage fault system handler enable 0b (R/W) = disabled 1b (R/W) = enabled
15	SVCALLPENDED	R	0h	SVCall pended flag. 0b (R/W) = not pended 1b (R/W) = pended
14	BUSFAULTPENDED	R	0h	BusFault pended flag. 0b (R/W) = not pended 1b (R/W) = pended
13	MEMFAULTPENDED	R	0h	MemManage pended flag. 0b (R/W) = not pended 1b (R/W) = pended
12	USGFAULTPENDED	R	0h	usage fault pended flag. 0b (R/W) = not pended 1b (R/W) = pended
11	SYSTICKACT	R	0h	SysTick active flag. 0b (R/W) = not active 1b (R/W) = active

**Table 2-68. SHCSR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
10	PENDSVACT	R	0h	PendSV active flag. 0b (R/W) = not active 1b (R/W) = active
9	RESERVED	R/W	0h	
8	MONITORACT	R	0h	the Monitor active flag. 0b (R/W) = not active 1b (R/W) = active
7	SVCALLACT	R	0h	SVCall active flag. 0b (R/W) = not active 1b (R/W) = active
6-4	RESERVED	R/W	0h	
3	USGFAULTACT	R	0h	UsageFault active flag. 0b (R/W) = not active 1b (R/W) = active
2	RESERVED	R/W	0h	
1	BUSFAULTACT	R	0h	BusFault active flag. 0b (R/W) = not active 1b (R/W) = active
0	MEMFAULTACT	R	0h	MemManage active flag. 0b (R/W) = not active 1b (R/W) = active

#### 2.4.5.11 CFSR Register (Offset = D28h) [reset = 00000000h]

CFSR is shown in [Figure 2-61](#) and described in [Table 2-69](#).

Configurable Fault Status Registers. Use the three Configurable Fault Status Registers to obtain information about local faults. These registers include: Memory Manage Fault Status Register (0xE000ED28), Bus Fault Status Register (0xE000ED29), Usage Fault Status Register (0xE000ED2A). The flags in these registers indicate the causes of local faults. Multiple flags can be set if more than one fault occurs. These register are read/write-clear. This means that they can be read normally, but writing a 1 to any bit clears that bit.

**Figure 2-61. CFSR Register**

31	30	29	28	27	26	25	24
RESERVED						DIVBYZERO	UNALIGNED
R/W-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED				NOCP	INVPC	INVSTATE	UNDEFINSTR
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
BFARVALID	RESERVED	LSPERR	STKERR	UNSTKERR	IMPRECISERR	PRECISERR	IBUSERR
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
MMARVALID	RESERVED	MLSPERR	MSTKERR	MUNSTKERR	RESERVED	DACCVIOL	IACCVIOL
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 2-69. CFSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R/W	0h	
25	DIVBYZERO	R/W	0h	When DIV_0_TRP (see Configuration Control Register) is enabled and an SDIV or UDIV instruction is used with a divisor of 0, this fault occurs. The instruction is executed and the return PC points to it. If DIV_0_TRP is not set, then the divide returns a quotient of 0.
24	UNALIGNED	R/W	0h	When UNALIGN_TRP is enabled (see Configuration Control Register), and there is an attempt to make an unaligned memory access, then this fault occurs. Unaligned LDM/STM/LDRD/STRD instructions always fault irrespective of the setting of UNALIGN_TRP.
23-20	RESERVED	R/W	0h	
19	NOCP	R/W	0h	Attempt to use a coprocessor instruction. The processor does not support coprocessor instructions.
18	INVPC	R/W	0h	Attempt to load EXC_RETURN into PC illegally. Invalid instruction, invalid context, invalid value. The return PC points to the instruction that tried to set the PC.
17	INVSTATE	R/W	0h	Invalid combination of EPSR and instruction, for reasons other than UNDEFINED instruction. Return PC points to faulting instruction, with the invalid state.
16	UNDEFINSTR	R/W	0h	The UNDEFINSTR flag is set when the processor attempts to execute an undefined instruction. This is an instruction that the processor cannot decode. The return PC points to the undefined instruction.
15	BFARVALID	R/W	0h	This bit is set if the Bus Fault Address Register (BFAR) contains a valid address. This is true after a bus fault where the address is known. Other faults can clear this bit, such as a Mem Manage fault occurring later. If a Bus fault occurs that is escalated to a Hard Fault because of priority, the Hard Fault handler must clear this bit. This prevents problems if returning to a stacked active Bus fault handler whose BFAR value has been overwritten.
14	RESERVED	R/W	0h	

**Table 2-69. CFSR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
13	LSPERR	R/W	0h	Indicates if bus fault occurred during FP lazy state preservation.
12	STKERR	R/W	0h	Stacking from exception has caused one or more bus faults. The SP is still adjusted and the values in the context area on the stack might be incorrect. The BFAR is not written.
11	UNSTKERR	R/W	0h	Unstack from exception return has caused one or more bus faults. This is chained to the handler, so that the original return stack is still present. SP is not adjusted from failing return and new save is not performed. The BFAR is not written.
10	IMPRECISERR	R/W	0h	Imprecise data bus error. It is a BusFault, but the Return PC is not related to the causing instruction. This is not a synchronous fault. So, if detected when the priority of the current activation is higher than the Bus Fault, it only pends. Bus fault activates when returning to a lower priority activation. If a precise fault occurs before returning to a lower priority exception, the handler detects both IMPRECISERR set and one of the precise fault status bits set at the same time. The BFAR is not written.
9	PRECISERR	R/W	0h	Precise data bus error return.
8	IBUSERR	R/W	0h	Instruction bus error flag. The IBUSERR flag is set by a prefetch error. The fault stops on the instruction, so if the error occurs under a branch shadow, no fault occurs. The BFAR is not written.
7	MMARVALID	R/W	0h	Memory Manage Address Register (MMAR) address valid flag. A later-arriving fault, such as a bus fault, can clear a memory manage fault.. If a MemManage fault occurs that is escalated to a Hard Fault because of priority, the Hard Fault handler must clear this bit. This prevents problems on return to a stacked active MemManage handler whose MMAR value has been overwritten.
6	RESERVED	R/W	0h	
5	MLSPERR	R/W	0h	Indicates if MemManage fault occurred during FP lazy state preservation.
4	MSTKERR	R/W	0h	Stacking from exception has caused one or more access violations. The SP is still adjusted and the values in the context area on the stack might be incorrect. The MMAR is not written.
3	MUNSTKERR	R/W	0h	Unstack from exception return has caused one or more access violations. This is chained to the handler, so that the original return stack is still present. SP is not adjusted from failing return and new save is not performed. The MMAR is not written.
2	RESERVED	R/W	0h	
1	DACCVIOL	R/W	0h	Data access violation flag. Attempting to load or store at a location that does not permit the operation sets the DACCVIOL flag. The return PC points to the faulting instruction. This error loads MMAR with the address of the attempted access.
0	IACCVIOL	R/W	0h	Instruction access violation flag. Attempting to fetch an instruction from a location that does not permit execution sets the IACCVIOL flag. This occurs on any access to an XN region, even when the MPU is disabled or not present. The return PC points to the faulting instruction. The MMAR is not written.

#### 2.4.5.12 HFSR Register (Offset = D2Ch) [reset = 00000000h]

HFSR is shown in [Figure 2-62](#) and described in [Table 2-70](#).

Hard Fault Status Register. Use the Hard Fault Status Register (HFSR) to obtain information about events that activate the Hard Fault handler. The HFSR is a write-clear register. This means that writing a 1 to a bit clears that bit

**Figure 2-62. HFSR Register**

31	30	29	28	27	26	25	24
DEBUGEVT	FORCED			RESERVED			
R/W-0h	R/W-0h			R/W-0h			
23	22	21	20	19	18	17	16
				RESERVED			
				R/W-0h			
15	14	13	12	11	10	9	8
				RESERVED			
				R/W-0h			
7	6	5	4	3	2	1	0
			RESERVED		VECTTBL		RESERVED
			R/W-0h		R/W-0h		R/W-0h

**Table 2-70. HFSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31	DEBUGEVT	R/W	0h	This bit is set if there is a fault related to debug. This is only possible when halting debug is not enabled. For monitor enabled debug, it only happens for BKPT when the current priority is higher than the monitor. When both halting and monitor debug are disabled, it only happens for debug events that are not ignored (minimally, BKPT). The Debug Fault Status Register is updated.
30	FORCED	R/W	0h	Hard Fault activated because a Configurable Fault was received and cannot activate because of priority or because the Configurable Fault is disabled. The Hard Fault handler then has to read the other fault status registers to determine cause.
29-2	RESERVED	R/W	0h	
1	VECTTBL	R/W	0h	This bit is set if there is a fault because of vector table read on exception processing (Bus Fault). This case is always a Hard Fault. The return PC points to the pre-empted instruction.
0	RESERVED	R/W	0h	

### 2.4.5.13 DFSR Register (Offset = D30h) [reset = 00000000h]

DFSR is shown in [Figure 2-63](#) and described in [Table 2-71](#).

Debug Fault Status Register. Use the Debug Fault Status Register to monitor: external debug requests, vector catches, data watchpoint match, BKPT instruction execution, halt requests. Multiple flags in the Debug Fault Status Register can be set when multiple fault conditions occur. The register is read/write clear. This means that it can be read normally. Writing a 1 to a bit clears that bit. Note that these bits are not set unless the event is caught. This means that it causes a stop of some sort. If halting debug is enabled, these events stop the processor into debug. If debug is disabled and the debug monitor is enabled, then this becomes a debug monitor handler call, if priority permits. If debug and the monitor are both disabled, some of these events are Hard Faults, and the DBGEVT bit is set in the Hard Fault status register, and some are ignored.

**Figure 2-63. DFSR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		EXTERNAL		VCATCH		DWTRAP	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

**Table 2-71. DFSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R/W	0h	
4	EXTERNAL	R/W	0h	External debug request flag. The processor stops on next instruction boundary. 0b (R/W) = EDBGREQ signal not asserted 1b (R/W) = EDBGREQ signal asserted
3	VCATCH	R/W	0h	Vector catch flag. When the VCATCH flag is set, a flag in one of the local fault status registers is also set to indicate the type of fault. 0b (R/W) = no vector catch occurred 1b (R/W) = vector catch occurred
2	DWTRAP	R/W	0h	Data Watchpoint and Trace (DWT) flag. The processor stops at the current instruction or at the next instruction. 0b (R/W) = no DWT match 1b (R/W) = DWT match
1	BKPT	R/W	0h	BKPT flag. The BKPT flag is set by a BKPT instruction in flash patch code, and also by normal code. Return PC points to breakpoint containing instruction. 0b (R/W) = no BKPT instruction execution 1b (R/W) = BKPT instruction execution
0	HALTED	R/W	0h	Halt request flag. The processor is halted on the next instruction. 0b (R/W) = no halt request 1b (R/W) = halt requested by NVIC, including step

#### 2.4.5.14 MMFAR Register (Offset = D34h)

MMFAR is shown in [Figure 2-64](#) and described in [Table 2-72](#).

Mem Manage Fault Address Register. Use the Memory Manage Fault Address Register to read the address of the location that caused a Memory Manage Fault.

**Figure 2-64. MMFAR Register**

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
ADDRESS																															
R/W-																															

**Table 2-72. MMFAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ADDRESS	R/W	Undefined	Mem Manage fault address field. ADDRESS is the data address of a faulted load or store attempt. When an unaligned access faults, the address is the actual address that faulted. Because an access can be split into multiple parts, each aligned, this address can be any offset in the range of the requested size. Flags in the Memory Manage Fault Status Register indicate the cause of the fault

### 2.4.5.15 BFAR Register (Offset = D38h)

BFAR is shown in [Figure 2-65](#) and described in [Table 2-73](#).

Bus Fault Address Register. Use the Bus Fault Address Register to read the address of the location that generated a Bus Fault.

**Figure 2-65. BFAR Register**

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
ADDRESS																															
R/W-																															

**Table 2-73. BFAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ADDRESS	R/W	Undefined	Bus fault address field. ADDRESS is the data address of a faulted load or store attempt. When an unaligned access faults, the address is the address requested by the instruction, even if that is not the address that faulted. Flags in the Bus Fault Status Register indicate the cause of the fault

#### 2.4.5.16 AFSR Register (Offset = D3Ch) [reset = 00000000h]

AFSR is shown in [Figure 2-66](#) and described in [Table 2-74](#).

Auxiliary Fault Status Register. Use the Auxiliary Fault Status Register (AFSR) to determine additional system fault information to software. The AFSR flags map directly onto the AUXFAULT inputs of the processor, and a single-cycle high level on an external pin causes the corresponding AFSR bit to become latched as one. The bit can only be cleared by writing a one to the corresponding AFSR bit. When an AFSR bit is written or latched as one, an exception does not occur. If you require an exception, you must use an interrupt.

**Figure 2-66. AFSR Register**

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
IMPDEF																															
R/W-0h																															

**Table 2-74. AFSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	IMPDEF	R/W	0h	Implementation defined. The bits map directly onto the signal assignment to the AUXFAULT inputs.

#### 2.4.5.17 PFR0 Register (Offset = D40h) [reset = 00000030h]

PFR0 is shown in [Figure 2-67](#) and described in [Table 2-75](#).

Processor Feature register0. Processor Feature register0

**Figure 2-67. PFR0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								STATE1				STATE0			
R-0h								R-3h				R-0h			

**Table 2-75. PFR0 Register Field Descriptions**

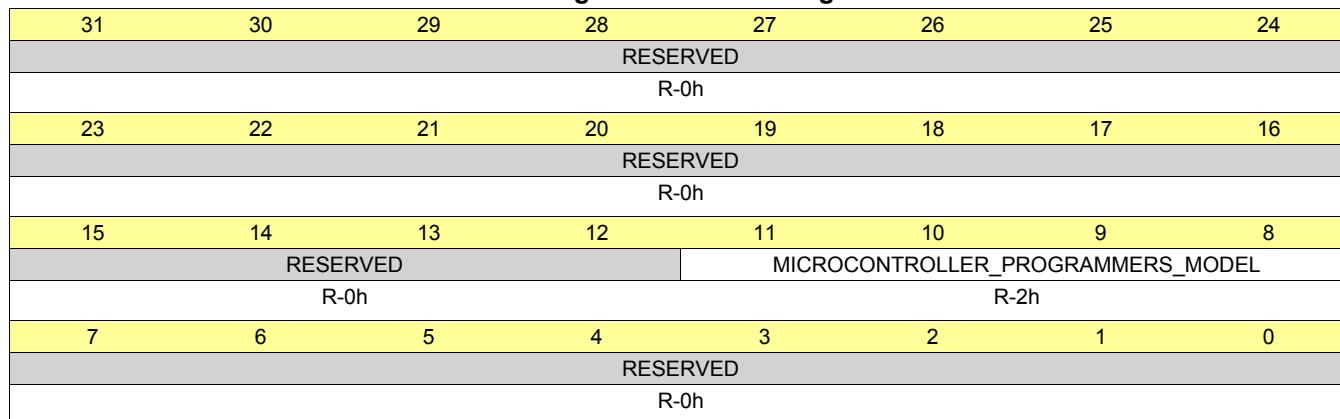
Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	
7-4	STATE1	R	3h	State1 (T-bit == 1) 0b (R/W) = N/A 1b (R/W) = N/A 10b (R/W) = Thumb-2 encoding with the 16-bit basic instructions plus 32-bit Buncond/BL but no other 32-bit basic instructions (Note non-basic 32-bit instructions can be added using the appropriate instruction attribute, but other 32-bit basic instructions cannot.) 11b (R/W) = Thumb-2 encoding with all Thumb-2 basic instructions
3-0	STATE0	R	0h	State0 (T-bit == 0) 0b (R/W) = no ARM encoding 1b (R/W) = N/A

#### 2.4.5.18 PFR1 Register (Offset = D44h) [reset = 00000200h]

PFR1 is shown in [Figure 2-68](#) and described in [Table 2-76](#).

Processor Feature register1. Processor Feature register1

**Figure 2-68. PFR1 Register**



**Table 2-76. PFR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	
11-8	MICROCONTROLLER_PROGRAMMERS_MODEL	R	2h	Microcontroller programmer's model 0b (R/W) = not supported 10b (R/W) = two-stack support
7-0	RESERVED	R	0h	

### 2.4.5.19 DFR0 Register (Offset = D48h) [reset = 00100000h]

DFR0 is shown in [Figure 2-69](#) and described in [Table 2-77](#).

Debug Feature register0. This register provides a high level view of the debug system. Further details are provided in the debug infrastructure itself.

**Figure 2-69. DFR0 Register**

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

**Table 2-77. DFR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	
23-20	MICROCONTROLLER_DEBUG_MODEL	R	1h	Microcontroller Debug Model - memory mapped 0b (R/W) = not supported 1b (R/W) = Microcontroller debug v1 (ITMv1, DWTv1, optional ETM)
19-0	RESERVED	R	0h	

#### 2.4.5.20 AFR0 Register (Offset = D4Ch) [reset = 00000000h]

AFR0 is shown in [Figure 2-70](#) and described in [Table 2-78](#).

Auxiliary Feature register0. RESERVED.

**Figure 2-70. AFR0 Register**

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

**Table 2-78. AFR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	RESERVED

### 2.4.5.21 MMFR0 Register (Offset = D50h) [reset = 00100030h]

MMFR0 is shown in [Figure 2-71](#) and described in [Table 2-79](#).

Memory Model Feature register0. General information on the memory model and memory management support.

**Figure 2-71. MMFR0 Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
AUILIARY_REGISTER_SUPPORT				RESERVED			
R-1h							
15	14	13	12	11	10	9	8
OUTER_NON_SHARABLE_SUPPORT				CACHE_COHERENCE_SUPPORT			
R-0h							
7	6	5	4	3	2	1	0
PMSA_SUPPORT				RESERVED			
R-3h							

**Table 2-79. MMFR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	
23-20	AUILIARY_REGISTER_SUPPORT	R	1h	Auxiliary register support 0b (R/W) = not supported 1b (R/W) = Auxiliary control register
19-16	RESERVED	R	0h	
15-12	OUTER_NON_SHARABLE_SUPPORT	R	0h	Outer non-shareable support 0b (R/W) = Outer non-shareable not supported 1b (R/W) = Outer shareable supported
11-8	CACHE_COHERENCE_SUPPORT	R	0h	Cache coherence support 0b (R/W) = no shared support 1b (R/W) = partial-inner-shared coherency (coherency amongst some - but not all - of the entities within an inner-coherent domain) 10b (R/W) = full-inner-shared coherency (coherency amongst all of the entities within an inner-coherent domain) 11b (R/W) = full coherency (coherency amongst all of the entities)
7-4	PMSA_SUPPORT	R	3h	PMSA support 0b (R/W) = not supported 1b (R/W) = IMPLEMENTATION DEFINED (N/A) 10b (R/W) = PMSA base (features as defined for ARMv6) (N/A) 11b (R/W) = PMSAv7 (base plus subregion support)
3-0	RESERVED	R	0h	

#### 2.4.5.22 MMFR1 Register (Offset = D54h) [reset = 00000000h]

MMFR1 is shown in [Figure 2-72](#) and described in [Table 2-80](#).

Memory Model Feature register1. General information on the memory model and memory management support.

**Figure 2-72. MMFR1 Register**

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

**Table 2-80. MMFR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	RESERVED

### 2.4.5.23 MMFR2 Register (Offset = D58h) [reset = 01000000h]

MMFR2 is shown in [Figure 2-73](#) and described in [Table 2-81](#).

Memory Model Feature register2. General information on the memory model and memory management support.

**Figure 2-73. MMFR2 Register**

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

**Table 2-81. MMFR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	
27-24	WAIT_FOR_INTERRUPT_STALLING	R	1h	wait for interrupt stalling 0b (R/W) = not supported 1b (R/W) = wait for interrupt supported
23-0	RESERVED	R	0h	

#### 2.4.5.24 MMFR3 Register (Offset = D5Ch) [reset = 00000000h]

MMFR3 is shown in [Figure 2-74](#) and described in [Table 2-82](#).

Memory Model Feature register3. General information on the memory model and memory management support.

**Figure 2-74. MMFR3 Register**

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

**Table 2-82. MMFR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	RESERVED	R	0h	RESERVED

### 2.4.5.25 ISAR0 Register (Offset = D60h) [reset = 01101110h]

ISAR0 is shown in Figure 2-75 and described in Table 2-83.

ISA Feature register0. Information on the instruction set attributes register

**Figure 2-75. ISAR0 Register**

31	30	29	28	27	26	25	24
RESERVED				DIVIDE_INSTRS			
R-0h				R-1h			
23	22	21	20	19	18	17	16
DEBUG_INSTRS				COPROC_INSTRS			
R-1h				R-0h			
15	14	13	12	11	10	9	8
CMPBRANCH_INSTRS				BITFIELD_INSTRS			
R-1h				R-1h			
7	6	5	4	3	2	1	0
BITCOUNT_INSTRS				RESERVED			
R-1h				R-0h			

**Table 2-83. ISAR0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	
27-24	DIVIDE_INSTRS	R	1h	Divide instructions 0b (R/W) = no divide instructions present 1b (R/W) = adds SDIV, UDIV (v1 quotient only result)
23-20	DEBUG_INSTRS	R	1h	Debug instructions 0b (R/W) = no debug instructions present 1b (R/W) = adds BKPT
19-16	COPROC_INSTRS	R	0h	Coprocessor instructions 0b (R/W) = no coprocessor support, other than for separately attributed architectures such as CP15 or VFP 1b (R/W) = adds generic CDP, LDC, MCR, MRC, STC 10b (R/W) = adds generic CDP2, LDC2, MCR2, MRC2, STC2 11b (R/W) = adds generic MCRR, MRRC 100b (R/W) = adds generic MCRR2, MRRC2
15-12	CMPBRANCH_INSTRS	R	1h	CmpBranch instructions 0b (R/W) = no combined compare-and-branch instructions present 1b (R/W) = adds CB{N}Z
11-8	BITFIELD_INSTRS	R	1h	BitField instructions 0b (R/W) = no bitfield instructions present 1b (R/W) = adds BFC, BFI, SBFX, UBFX
7-4	BITCOUNT_INSTRS	R	1h	BitCount instructions 0b (R/W) = no bit-counting instructions present 1b (R/W) = adds CLZ
3-0	RESERVED	R	0h	

#### 2.4.5.26 ISAR1 Register (Offset = D64h) [reset = 02112000h]

ISAR1 is shown in Figure 2-76 and described in Table 2-84.

ISA Feature register1. Information on the instruction set attributes register

**Figure 2-76. ISAR1 Register**

31	30	29	28	27	26	25	24
RESERVED				INTERWORK_INSTRS			
R-0h				R-2h			
23	22	21	20	19	18	17	16
IMMEDIATE_INSTRS				IFTHEN_INSTRS			
R-1h				R-1h			
15	14	13	12	11	10	9	8
ETEND_INSRS				RESERVED			
R-2h				R-0h			
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

**Table 2-84. ISAR1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	
27-24	INTERWORK_INSTRS	R	2h	Interwork instructions 0b (R/W) = no interworking instructions supported 1b (R/W) = adds BX (and T bit in PSRs) 10b (R/W) = adds BLX, and PC loads have BX-like behavior 11b (R/W) = N/A
23-20	IMMEDIATE_INSTRS	R	1h	Immediate instructions 0b (R/W) = no special immediate-generating instructions present 1b (R/W) = adds ADDW, MOVW, MOVT, SUBW
19-16	IFTHEN_INSTRS	R	1h	IfThen instructions 0b (R/W) = IT instructions not present 1b (R/W) = adds IT instructions (and IT bits in PSRs)
15-12	ETEND_INSRS	R	2h	Extend instructions. Note that the shift options on these instructions are also controlled by the WithShifts_instrs attribute. 0b (R/W) = no scalar (i.e. non-SIMD) sign/zero-extend instructions present 1b (R/W) = adds SXTB, SXTH, UXTB, UXTH 10b (R/W) = N/A
11-0	RESERVED	R	0h	

### 2.4.5.27 ISAR2 Register (Offset = D68h) [reset = 21232231h]

ISAR2 is shown in [Figure 2-77](#) and described in [Table 2-85](#).

ISA Feature register2. Information on the instruction set attributes register

**Figure 2-77. ISAR2 Register**

31	30	29	28	27	26	25	24
REVERSAL_INSTRS				RESERVED			
R-2h				R-1h			
23	22	21	20	19	18	17	16
MULTU_INSTRS				MULTS_INSTRS			
R-2h				R-3h			
15	14	13	12	11	10	9	8
MULT_INSTRS				MULTIACCESSINT_INSTRS			
R-2h				R-2h			
7	6	5	4	3	2	1	0
MEMHINT_INSTRS				LOADSTORE_INSTRS			
R-3h				R-1h			

**Table 2-85. ISAR2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	REVERSAL_INSTRS	R	2h	Reversal instructions 0b (R/W) = no reversal instructions present 1b (R/W) = adds REV, REV16, REVSH 10b (R/W) = adds RBIT
27-24	RESERVED	R	1h	
23-20	MULTU_INSTRS	R	2h	Multiply instructions (advanced, unsigned) 0b (R/W) = no unsigned multiply instructions present 1b (R/W) = adds UMULL, UMLAL 10b (R/W) = N/A
19-16	MULTS_INSTRS	R	3h	Multiply instructions (advanced, signed) 0b (R/W) = no signed multiply instructions present 1b (R/W) = adds SMULL, SMLAL 10b (R/W) = N/A 11b (R/W) = N/A
15-12	MULT_INSTRS	R	2h	Multiply instructions 0b (R/W) = only MUL present 1b (R/W) = adds MLA 10b (R/W) = adds MLS
11-8	MULTIACCESSINT_INST_RS	R	2h	Multi-Access interruptible instructions 0b (R/W) = the (LDM/STM) instructions are non-interruptible 1b (R/W) = the (LDM/STM) instructions are restartable 10b (R/W) = the (LDM/STM) instructions are continuatable
7-4	MEMHINT_INSTRS	R	3h	MemoryHint instructions 0b (R/W) = no memory hint instructions present 1b (R/W) = adds PLD 10b (R/W) = adds PLD (ie a repeat on value 1) 11b (R/W) = adds PLI
3-0	LOADSTORE_INSTRS	R	1h	LoadStore instructions 0b (R/W) = no additional normal load/store instructions present 1b (R/W) = adds LDRD/STRD

#### 2.4.5.28 ISAR3 Register (Offset = D6Ch) [reset = 01111131h]

ISAR3 is shown in [Figure 2-78](#) and described in [Table 2-86](#).

ISA Feature register3. Information on the instruction set attributes register

**Figure 2-78. ISAR3 Register**

31	30	29	28	27	26	25	24
RESERVED				TRUE NOP_INSTRS			
R-0h				R-1h			
23	22	21	20	19	18	17	16
THUMBCOPY_INSTRS				TABBRANCH_INSTRS			
R-1h				R-1h			
15	14	13	12	11	10	9	8
SYNCPRIM_INSTRS				SVC_INSTRS			
R-1h				R-1h			
7	6	5	4	3	2	1	0
SIMD_INSTRS				SATRUATE_INSTRS			
R-3h				R-1h			

**Table 2-86. ISAR3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	
27-24	TRUE NOP_INSTRS	R	1h	TrueNOP instructions 0b (R/W) = true NOP instructions not present - that is, NOP instructions with no register dependencies 1b (R/W) = adds "true NOP", and the capability of additional "NOP compatible hints"
23-20	THUMBCOPY_INSTRS	R	1h	ThumbCopy instructions 0b (R/W) = Thumb MOV(register) instruction does not allow low reg -> low reg 1b (R/W) = adds Thumb MOV(register) low reg -> low reg and the CPY alias
19-16	TABBRANCH_INSTRS	R	1h	TableBranch instructions 0b (R/W) = no table-branch instructions present 1b (R/W) = adds TBB, TBH
15-12	SYNCPRIM_INSTRS	R	1h	SyncPrim instructions. Note there are no LDREXD or STREXD in ARMv7-M. This attribute is used in conjunction with the SyncPrim_instrs_frac attribute in ID_ISAR4[23:20]. 0b (R/W) = no synchronization primitives present 1b (R/W) = adds LDREX, STREX 10b (R/W) = adds LDREXB, LDREXH, LDREXD, STREXB, STREXH, STREXD, CLREX(N/A)
11-8	SVC_INSTRS	R	1h	SVC instructions 0b (R/W) = no SVC (SWI) instructions present 1b (R/W) = adds SVC (SWI)
7-4	SIMD_INSTRS	R	3h	SIMD instructions 0b (R/W) = no SIMD instructions present 1b (R/W) = adds SSAT, USAT (and the Q flag in the PSRs) 11b (R/W) = N/A
3-0	SATRUATE_INSTRS	R	1h	Saturate instructions 0b (R/W) = no non-SIMD saturate instructions present 1b (R/W) = N/A

### 2.4.5.29 ISAR4 Register (Offset = D70h) [reset = 01310132h]

ISAR4 is shown in Figure 2-79 and described in Table 2-87.

ISA Feature register4. Information on the instruction set attributes register

**Figure 2-79. ISAR4 Register**

31	30	29	28	27	26	25	24
RESERVED				PSR_M_INSTRS			
R-0h				R-1h			
23	22	21	20	19	18	17	16
SYNCPRIM_INSTRS_FRAC				BARRIER_INSTRS			
R-3h				R-1h			
15	14	13	12	11	10	9	8
RESERVED				WRITEBACK_INSTRS			
R-0h				R-1h			
7	6	5	4	3	2	1	0
WITHSHIFTS_INSTRS				UNPRIV_INSTRS			
R-0h				R-2h			

**Table 2-87. ISAR4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	
27-24	PSR_M_INSTRS	R	1h	PSR_M_instrs 0b (R/W) = instructions not present 1b (R/W) = adds CPS, MRS, and MSR instructions (M-profile forms)
23-20	SYNCPRIM_INSTRS_FRAC	R	3h	SyncPrim_instrs_frac 0b (R/W) = no additional support 11b (R/W) = adds CLREX, LDREXB, STREXB, LDREXH, STREXH
19-16	BARRIER_INSTRS	R	1h	Barrier instructions 0b (R/W) = no barrier instructions supported 1b (R/W) = adds DMB, DSB, ISB barrier instructions
15-12	RESERVED	R	0h	
11-8	WRITEBACK_INSTRS	R	1h	Writeback instructions 0b (R/W) = only non-writeback addressing modes present, except that LDMIA/STMDB/PUSH/POP instructions support writeback addressing. 1b (R/W) = adds all currently-defined writeback addressing modes (ARMv7, Thumb-2)
7-4	WITHSHIFTS_INSTRS	R	3h	WithShift instructions. Note that all additions only apply in cases where the encoding supports them; for example, there is no difference between levels 3 and 4 in the Thumb-2 instruction set. MOV instructions with shift options should instead be treated as ASR, LSL, LSR, ROR or RRX instructions. 000b (R/W) = non-zero shifts only support MOV and shift instructions 001b (R/W) = shifts of loads/stores over the range LSL 0-3 011b (R/W) = adds other constant shift options. 100b (R/W) = adds register-controlled shift options.
3-0	UNPRIV_INSTRS	R	2h	Unprivileged instructions 00b (R/W) = no "T variant" instructions exist 01b (R/W) = adds LDRBT, LDRT, STRBT, STRT 10b (R/W) = adds LDRHT, LDRSBT, LDRSHT, STRHT

#### 2.4.5.30 CPACR Register (Offset = D88h) [reset = 00F00000h]

CPACR is shown in [Figure 2-80](#) and described in [Table 2-88](#).

Coprocessor Access Control Register. The Coprocessor Access Control Register (CPACR) specifies the access privileges for coprocessors.

**Figure 2-80. CPACR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED					CP11		CP10		RESERVED						
R/W-0					R/W-3h			R/W-3h			R/W-0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															R/W-0

**Table 2-88. CPACR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R/W	0h	
23-22	CP11	R/W	3h	Access privileges for coprocessor 11. The possible values of each field are: 00b = Access denied. Any attempted access generates a NOCP UsageFault. 01b = Privileged access only. An unprivileged access generates a NOCP UsageFault. 10b = Reserved 11b = Full access Used in conjunction with the control for CP10, this controls access to the Floating Point Coprocessor.
21-20	CP10	R/W	3h	Access privileges for coprocessor 10. The possible values of each field are: 00b = Access denied. Any attempted access generates a NOCP UsageFault. 01b = Privileged access only. An unprivileged access generates a NOCP UsageFault. 10b = Reserved 11b = Full access Used in conjunction with the control for CP11, this controls access to the Floating Point Coprocessor.
19-0	RESERVED	R/W	0h	

## 2.4.6 SCnSCB Registers

Table 2-89 lists the memory-mapped registers for the SCnSCB. All register offset addresses not listed in Table 2-89 should be considered as reserved locations and the register contents should not be modified.

**Table 2-89. SCnSCB Registers**

Offset	Acronym	Register Name	Type	Reset	Section
4h	ICTR	Interrupt Control Type Register	read-only	00000001h	Section 2.4.6.1
8h	ACTLR	Auxiliary Control Register	read-write	00000000h	Section 2.4.6.2

#### 2.4.6.1 ICTR Register (Offset = 4h) [reset = 00000001h]

ICTR is shown in [Figure 2-81](#) and described in [Table 2-90](#).

Interrupt Control Type Register. Read the Interrupt Controller Type Register to see the number of interrupt lines that the NVIC supports.

**Figure 2-81. ICTR Register**

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

**Table 2-90. ICTR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	
4-0	INTLINESNUM	R	1h	Total number of interrupt lines in groups of 32. 0000b = 0 to 32 0001b = 33 to 64 0010b = 65 to 96 0011b = 97 to 128 0100b = 129 to 160 0101b = 161 to 192 0110b = 193 to 224 0111b = 225 to 256

#### 2.4.6.2 ACTLR Register (Offset = 8h) [reset = 00000000h]

ACTLR is shown in [Figure 2-82](#) and described in [Table 2-91](#).

Auxiliary Control Register. Use the Auxiliary Control Register to disable certain aspects of functionality within the processor

**Figure 2-82. ACTLR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED						DISOOFP	DISFPCA
R/W-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED				DISFOLD	DISDEFWBUF	DISMCYCINT	
R/W-0h				R/W-0h	R/W-0h	R/W-0h	

**Table 2-91. ACTLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-10	RESERVED	R/W	0h	Reserved
9	DISOOFP	R/W	0h	Disables floating point instructions completing out of order with respect to integer instructions.
8	DISFPCA	R/W	0h	Disables automatic update of CONTROL.FPCA.
7-3	RESERVED	R/W	0h	Reserved
2	DISFOLD	R/W	0h	Disables IT folding.
1	DISDEFWBUF	R/W	0h	Disables write buffer us during default memory map accesses. This causes all bus faults to be precise bus faults but decreases the performance of the processor because the stores to memory have to complete before the next instruction can be executed.
0	DISMCYCINT	R/W	0h	Disables interruption of multi-cycle instructions. This increases the interrupt latency of the processor because LDM/STM completes before interrupt stacking occurs.

## 2.4.7 COREDEBUG Registers

**Table 2-92** lists the memory-mapped registers for the COREDEBUG. All register offset addresses not listed in **Table 2-92** should be considered as reserved locations and the register contents should not be modified.

**Table 2-92. COREDEBUG Registers**

Offset	Acronym	Register Name	Type	Reset	Section
DF0h	DHCSR	Debug Halting Control and Status Register	read-write	00000000h	<a href="#">Section 2.4.7.1</a>
DF4h	DCRSR	Deubg Core Register Selector Register	write-only	Undefined	<a href="#">Section 2.4.7.2</a>
DF8h	DCRDR	Debug Core Register Data Register	read-write	Undefined	<a href="#">Section 2.4.7.3</a>
DFCh	DEMCR	Debug Exception and Monitor Control Register	read-write	00000000h	<a href="#">Section 2.4.7.4</a>

### 2.4.7.1 DHCSR Register (Offset = DF0h) [reset = 00000000h]

Register mask: FFFEFFFFh

DHCSR is shown in [Figure 2-83](#) and described in [Table 2-93](#).

Debug Halting Control and Status Register. The purpose of the Debug Halting Control and Status Register (DHCSR) is to provide status information about the state of the processor, enable core debug, halt and step the processor. For writes, 0xA05F must be written to bits [31:16], otherwise the write operation is ignored and no bits are written into the register. If not enabled for Halting mode, C\_DEBUGEN = 1, all other fields are disabled. This register is not reset on a system reset. It is reset by a POR reset. However, the C\_HALT bit always clears on a system reset. To halt on a reset, the following bits must be enabled: bit [0], VC\_CORERESET, of the Debug Exception and Monitor Control Register and bit [0], C\_DEBUGEN, of the Debug Halting Control and Status Register. Note that writes to this register in any size other than word are Unpredictable. It is acceptable to read in any size, and you can use it to avoid or intentionally change a sticky bit. Bit 16 of DHCSR is Unpredictable on reset.

**Figure 2-83. DHCSR Register**

31	30	29	28	27	26	25	24
RESERVED					S_RESET_ST	S_RETIRE_ST	
R/W-0h					R-0h	R-0h	
23	22	21	20	19	18	17	16
RESERVED			S_LOCKUP	S_SLEEP	S_HALT	S_REGRDY	
R/W-0h			R-0h	R-0h	R-0h	R-X	
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		C_SNAPSTALL	RESERVED	C_MASKINTS	C_STEP	C_HALT	C_DEBUGEN
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 2-93. DHCSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R/W	0h	
25	S_RESET_ST	R	0h	Indicates that the core has been reset, or is now being reset, since the last time this bit was read. This a sticky bit that clears on read. So, reading twice and getting 1 then 0 means it was reset in the past. Reading twice and getting 1 both times means that it is being reset now (held in reset still).
24	S_RETIRE_ST	R	0h	Indicates that an instruction has completed since last read. This is a sticky bit that clears on read. This determines if the core is stalled on a load/store or fetch.
23-20	RESERVED	R/W	0h	
19	S_LOCKUP	R	0h	Reads as one if the core is running (not halted) and a lockup condition is present.
18	S_SLEEP	R	0h	Indicates that the core is sleeping (WFI, WFE, or SLEEP-ON-EXIT). Must use C_HALT to gain control or wait for interrupt to wake-up.
17	S_HALT	R	0h	The core is in debug state when S_HALT is set.
16	S_REGRDY	R	X	Register Read/Write on the Debug Core Register Selector register is available. Last transfer is complete.
15-6	RESERVED	R/W	0h	

**Table 2-93. DHCSR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	C_SNAPSTALL	R/W	0h	If the core is stalled on a load/store operation the stall ceases and the instruction is forced to complete. This enables Halting debug to gain control of the core. It can only be set if: C_DEBUGEN = 1 and C_HALT = 1. The core reads S_RETIRE_ST as 0. This indicates that no instruction has advanced. This prevents misuse. The bus state is Unpredictable when this is used. S_RETIRE can detect core stalls on load/store operations.
4	RESERVED	R/W	0h	
3	C_MASKINTS	R/W	0h	Mask interrupts when stepping or running in halted debug. Does not affect NMI, which is not maskable. Must only be modified when the processor is halted ( $S\_HALT == 1$ ). Also does not affect fault exceptions and SVC caused by execution of the instructions. CMASKINTS must be set or cleared before halt is released. This means that the writes to set or clear C_MASKINTS and to set or clear C_HALT must be separate.
2	C_STEP	R/W	0h	Steps the core in halted debug. When C_DEBUGEN = 0, this bit has no effect. Must only be modified when the processor is halted ( $S\_HALT == 1$ ).
1	C_HALT	R/W	0h	Halts the core. This bit is set automatically when the core Halts. For example Breakpoint. This bit clears on core reset. This bit can only be written if C_DEBUGEN is 1, otherwise it is ignored. When setting this bit to 1, C_DEBUGEN must also be written to 1 in the same value (value[1:0] is 2'b11). The core can halt itself, but only if C_DEBUGEN is already 1 and only if it writes with b11).
0	C_DEBUGEN	R/W	0h	Enables debug. This can only be written by AHB-AP and not by the core. It is ignored when written by the core, which cannot set or clear it. The core must write a 1 to it when writing C_HALT to halt itself.

#### 2.4.7.2 DCRSR Register (Offset = DF4h) [reset = Unknown]

DCRSR is shown in [Figure 2-84](#) and described in [Table 2-94](#).

Deubg Core Register Selector Register. The purpose of the Debug Core Register Selector Register (DCRSR) is to select the processor register to transfer data to or from. This write-only register generates a handshake to the core to transfer data to or from Debug Core Register Data Register and the selected register. Until this core transaction is complete, bit [16], S\_REGRDY, of the DHCSR is 0. Note that writes to this register in any size but word are Unpredictable. Note that PSR registers are fully accessible this way, whereas some read as 0 when using MRS instructions. Note that all bits can be written, but some combinations cause a fault when execution is resumed. Note that IT might be written and behaves as though in an IT block.

**Figure 2-84. DCRSR Register**

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

**Table 2-94. DCRSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-17	RESERVED	W	Undefined	
16	REGWRN	W	Undefined	Write = 1, Read = 0
15-5	RESERVED	W	Undefined	
4-0	REGSEL	W	Undefined	Register select 0b (R/W) = R0 1b (R/W) = R1 10b (R/W) = R2 11b (R/W) = R3 100b (R/W) = R4 101b (R/W) = R5 110b (R/W) = R6 111b (R/W) = R7 1000b (R/W) = R8 1001b (R/W) = R9 1010b (R/W) = R10 1011b (R/W) = R11 1100b (R/W) = R12 1101b (R/W) = Current SP 1110b (R/W) = LR 1111b (R/W) = DebugReturnAddress 10000b (R/W) = xPSR flags, execution state information, and exception number 10001b (R/W) = MSP (Main SP) 10010b (R/W) = PSP (Process SP) 10100b (R/W) = CONTROL bits [31:24], FAULTMASK bits [23:16], BASEPRI bits [15:8], PRIMASK bits [7:0]

#### 2.4.7.3 DCRDR Register (Offset = DF8h)

DCRDR is shown in [Figure 2-85](#) and described in [Table 2-95](#).

Debug Core Register Data Register. The purpose of the Debug Core Register Data Register (DCRDR) is to hold data for reading and writing registers to and from the processor. This is the data value written to the register selected by the Debug Register Selector Register. When the processor receives a request from the Debug Core Register Selector, this register is read or written by the processor using a normal load-store unit operation. If core register transfers are not being performed, software-based debug monitors can use this register for communication in non-halting debug. For example, OS RSD and Real View Monitor. This enables flags and bits to acknowledge state and indicate if commands have been accepted to, replied to, or accepted and replied to.

**Figure 2-85. DCRDR Register**

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

**Table 2-95. DCRDR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	DBGTMP	R/W	Undefined	Data temporary cache, for reading and writing registers

#### 2.4.7.4 DEMCR Register (Offset = DFCh) [reset = 00000000h]

DEMCR is shown in [Figure 2-86](#) and described in [Table 2-96](#).

Debug Exception and Monitor Control Register. The purpose of the Debug Exception and Monitor Control Register (DEMCR) is Vector catching and Debug monitor control. This register manages exception behavior under debug. Vector catching is only available to halting debug. The upper halfword is for monitor controls and the lower halfword is for halting exception support. This register is not reset on a system reset. This register is reset by a POR reset. Bits [19:16] are always cleared on a core reset. The debug monitor is enabled by software in the reset handler or later, or by the AHB-AP port. Vector catching is semi-synchronous. When a matching event is seen, a Halt is requested. Because the processor can only halt on an instruction boundary, it must wait until the next instruction boundary. As a result, it stops on the first instruction of the exception handler. However, two special cases exist when a vector catch has triggered: 1. If a fault is taken during a vector read or stack push error the halt occurs on the corresponding fault handler for the vector error or stack push. 2. If a late arriving interrupt detected during a vector read or stack push error it is not taken. That is, an implementation that supports the late arrival optimization must suppress it in this case.

**Figure 2-86. DEMCR Register**

31	30	29	28	27	26	25	24
RESERVED							TRCENA
R/W-0h							R/W-0h
23	22	21	20	19	18	17	16
RESERVED				MON_REQ	MON_STEP	MON_PEND	MON_EN
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED				VC_HARDERR	VC_INTERR	VC_BUSERR	
R/W-0h				R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
VC_STATERR	VC_CHKERR	VC_NOCPERR	VC_MMERR	RESERVED			VC_COREREST
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			R/W-0h

**Table 2-96. DEMCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R/W	0h	
24	TRCENA	R/W	0h	This bit must be set to 1 to enable use of the trace and debug blocks: Data Watchpoint and Trace (DWT), Instrumentation Trace Macrocell (ITM), Embedded Trace Macrocell (ETM), Trace Port Interface Unit (TPIU). This enables control of power usage unless tracing is required. The application can enable this, for ITM use, or use by a debugger. Note that if no debug or trace components are present in the implementation then it is not possible to set TRCENA.
23-20	RESERVED	R/W	0h	
19	MON_REQ	R/W	0h	This enables the monitor to identify how it wakes up. This bit clears on a Core Reset. 0b (R/W) = woken up by debug exception. 1b (R/W) = woken up by MON_PEND
18	MON_STEP	R/W	0h	When MON_EN = 1, this steps the core. When MON_EN = 0, this bit is ignored. This is the equivalent to C_STEP. Interrupts are only stepped according to the priority of the monitor and settings of PRIMASK, FAULTMASK, or BASEPRI.
17	MON_PEND	R/W	0h	Pend the monitor to activate when priority permits. This can wake up the monitor through the AHB-AP port. It is the equivalent to C_HALT for Monitor debug. This register does not reset on a system reset. It is only reset by a POR reset. Software in the reset handler or later, or by the DAP must enable the debug monitor.

**Table 2-96. DEMCR Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
16	MON_EN	R/W	0h	Enable the debug monitor. When enabled, the System handler priority register controls its priority level. If disabled, then all debug events go to Hard fault. C_DEBUGEN in the Debug Halting Control and Statue register overrides this bit. Vector catching is semi-synchronous. When a matching event is seen, a Halt is requested. Because the processor can only halt on an instruction boundary, it must wait until the next instruction boundary. As a result, it stops on the first instruction of the exception handler. However, two special cases exist when a vector catch has triggered: 1. If a fault is taken during vectoring, vector read or stack push error, the halt occurs on the corresponding fault handler, for the vector error or stack push. 2. If a late arriving interrupt comes in during vectoring, it is not taken. That is, an implementation that supports the late arrival optimization must suppress it in this case.
15-11	RESERVED	R/W	0h	
10	VC_HARDERR	R/W	0h	Debug trap on Hard Fault.
9	VC_INTERR	R/W	0h	Debug Trap on interrupt/exception service errors. These are a subset of other faults and catches before BUSERR or HARDERR.
8	VC_BUSERR	R/W	0h	Debug Trap on normal Bus error.
7	VC_STATERR	R/W	0h	Debug trap on Usage Fault state errors.
6	VC_CHKERR	R/W	0h	Debug trap on Usage Fault enabled checking errors.
5	VC_NOCPERR	R/W	0h	Debug trap on Usage Fault access to Coprocessor that is not present or marked as not present in CAR register.
4	VC_MMERR	R/W	0h	Debug trap on Memory Management faults.
3-1	RESERVED	R/W	0h	
0	VC_CORERESET	R/W	0h	Reset Vector Catch. Halt running system if Core reset occurs.

## 2.5 Debug Peripherals Registers

This section lists the Cortex-M4 Debug Peripheral FPB, DWT, ITM registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000\_E000.

---

**NOTE:** Register spaces that are not used are reserved for future or internal use. Software should not modify any reserved memory address.

---

### 2.5.1 FPB Registers

Table 2-97 lists the memory-mapped registers for the FPB. All register offset addresses not listed in Table 2-97 should be considered as reserved locations and the register contents should not be modified.

**Table 2-97. FPB Registers**

Offset	Acronym	Register Name	Type	Reset	Section
0h	FP_CTRL	Flash Patch Control Register	read-write	00000260h	<a href="#">Section 2.5.1.1</a>
4h	FP_REMAP	Flash Patch Remap Register	read-write	20000000h	<a href="#">Section 2.5.1.2</a>
8h	FP_COMP0	Flash Patch Comparator Registers	read-write	00000000h	<a href="#">Section 2.5.1.3</a>
Ch	FP_COMP1	Flash Patch Comparator Registers	read-write	00000000h	<a href="#">Section 2.5.1.4</a>
10h	FP_COMP2	Flash Patch Comparator Registers	read-write	00000000h	<a href="#">Section 2.5.1.5</a>
14h	FP_COMP3	Flash Patch Comparator Registers	read-write	00000000h	<a href="#">Section 2.5.1.6</a>
18h	FP_COMP4	Flash Patch Comparator Registers	read-write	00000000h	<a href="#">Section 2.5.1.7</a>
1Ch	FP_COMP5	Flash Patch Comparator Registers	read-write	00000000h	<a href="#">Section 2.5.1.8</a>
20h	FP_COMP6	Flash Patch Comparator Registers	read-write	00000000h	<a href="#">Section 2.5.1.9</a>
24h	FP_COMP7	Flash Patch Comparator Registers	read-write	00000000h	<a href="#">Section 2.5.1.10</a>

### 2.5.1.1 FP\_CTRL Register (Offset = 0h) [reset = 00000260h]

FP\_CTRL is shown in [Figure 2-87](#) and described in [Table 2-98](#).

Flash Patch Control Register. Use the Flash Patch Control Register to enable the flash patch block.

**Figure 2-87. FP\_CTRL Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED		NUM_CODE2		NUM_LIT			
R/W-0h		R-0h		R-2h			
7	6	5	4	3	2	1	0
NUM_CODE1				RESERVED		KEY	ENABLE
R-6h				R/W-0h		W-0h	R/W-0h

**Table 2-98. FP\_CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-14	RESERVED	R/W	0h	
13-12	NUM_CODE2	R	0h	Number of full banks of code comparators, sixteen comparators per bank. Where less than sixteen code comparators are provided, the bank count is zero, and the number present indicated by NUM_CODE. This read only field contains 3'b000 to indicate 0 banks for Cortex-M4 processor.
11-8	NUM_LIT	R	2h	Number of literal slots field. 0b (R/W) = no literal slots 10b (R/W) = two literal slots
7-4	NUM_CODE1	R	6h	Number of code slots field. 0b (R/W) = no code slots 10b (R/W) = two code slots 110b (R/W) = six code slots
3-2	RESERVED	R/W	0h	
1	KEY	W	0h	Key field. To write to the Flash Patch Control Register, you must write a 1 to this write-only bit.
0	ENABLE	R/W	0h	Flash patch unit enable bit 0b (R/W) = flash patch unit disabled 1b (R/W) = flash patch unit enabled

### 2.5.1.2 FP\_REMAP Register (Offset = 4h) [reset = 20000000h]

FP\_REMAP is shown in [Figure 2-88](#) and described in [Table 2-99](#).

Flash Patch Remap Register. Use the Flash Patch Remap Register to provide the location in System space where a matched address is remapped. The REMAP address is 8-word aligned, with one word allocated to each of the eight FPB comparators. A comparison match remaps to: {3'b001, REMAP, COMP[2:0], HADDR[1:0]} where: 3'b001 hardwires the remapped access to system space, REMAP is the 24-bit, 8-word aligned remap address, COMP is the matching comparator, HADDR[1:0] is the two Least Significant Bits (LSBs) of the original address. HADDR[1:0] is always 2'b00 for instruction fetches.

**Figure 2-88. FP\_REMAP Register**

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
RESERVE D	REMAP																								RESERVED						
R/W-	R/W-																								R/W-						

**Table 2-99. FP\_REMAP Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-29	RESERVED	R/W	1h	
28-5	REMAP	R/W	0h	Remap base address field.
4-0	RESERVED	R/W	0h	

### 2.5.1.3 FP\_COMP0 Register (Offset = 8h) [reset = 00000000h]

Register mask: 00000001h

FP\_COMP0 is shown in [Figure 2-89](#) and described in [Table 2-100](#).

Flash Patch Comparator Registers. Use the Flash Patch Comparator Registers to store the values to compare with the PC address.

**Figure 2-89. FP\_COMP0 Register**

31	30	29	28	27	26	25	24
REPLACE		RESERVED			COMP		
R/W-X		R/W-X			R/W-X		
23	22	21	20	19	18	17	16
			COMP				
			R/W-X				
15	14	13	12	11	10	9	8
			COMP				
			R/W-X				
7	6	5	4	3	2	1	0
			COMP		RESERVED	ENABLE	
			R/W-X		R/W-X	R/W-0h	

**Table 2-100. FP\_COMP0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	REPLACE	R/W	X	This selects what happens when the COMP address is matched. Settings other than b00 are only valid for instruction comparators. Literal comparators ignore non-b00 settings. Address remapping only takes place for the b00 setting. 0b (R/W) = remap to remap address. See FP_REMAP 1b (R/W) = set BKPT on lower halfword, upper is unaffected 10b (R/W) = set BKPT on upper halfword, lower is unaffected 11b (R/W) = set BKPT on both lower and upper halfwords.
29	RESERVED	R/W	X	
28-2	COMP	R/W	X	Comparison address.
1	RESERVED	R/W	X	
0	ENABLE	R/W	0h	Compare and remap enable for Flash Patch Comparator Register 0. The ENABLE bit of FP_CTRL must also be set to enable comparisons. Reset clears the ENABLE bit. 0b (R/W) = Flash Patch Comparator Register 0 compare and remap disabled 1b (R/W) = Flash Patch Comparator Register 0 compare and remap enabled

### 2.5.1.4 FP\_COMP1 Register (Offset = Ch) [reset = 00000000h]

Register mask: 00000001h

FP\_COMP1 is shown in [Figure 2-90](#) and described in [Table 2-101](#).

Flash Patch Comparator Registers. Use the Flash Patch Comparator Registers to store the values to compare with the PC address.

**Figure 2-90. FP\_COMP1 Register**

31	30	29	28	27	26	25	24
REPLACE		RESERVED			COMP		
R/W-X		R/W-X			R/W-X		
23	22	21	20	19	18	17	16
			COMP				
			R/W-X				
15	14	13	12	11	10	9	8
			COMP				
			R/W-X				
7	6	5	4	3	2	1	0
			COMP		RESERVED	ENABLE	
			R/W-X		R/W-X	R/W-0h	

**Table 2-101. FP\_COMP1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	REPLACE	R/W	X	This selects what happens when the COMP address is matched. Settings other than b00 are only valid for instruction comparators. Literal comparators ignore non-b00 settings. Address remapping only takes place for the b00 setting. 0b (R/W) = remap to remap address. See FP_REMAP 1b (R/W) = set BKPT on lower halfword, upper is unaffected 10b (R/W) = set BKPT on upper halfword, lower is unaffected 11b (R/W) = set BKPT on both lower and upper halfwords.
29	RESERVED	R/W	X	
28-2	COMP	R/W	X	Comparison address.
1	RESERVED	R/W	X	
0	ENABLE	R/W	0h	Compare and remap enable for Flash Patch Comparator Register 1. The ENABLE bit of FP_CTRL must also be set to enable comparisons. Reset clears the ENABLE bit. 0b (R/W) = Flash Patch Comparator Register 1 compare and remap disabled 1b (R/W) = Flash Patch Comparator Register 1 compare and remap enabled

### 2.5.1.5 FP\_COMP2 Register (Offset = 10h) [reset = 00000000h]

Register mask: 00000001h

FP\_COMP2 is shown in [Figure 2-91](#) and described in [Table 2-102](#).

Flash Patch Comparator Registers. Use the Flash Patch Comparator Registers to store the values to compare with the PC address.

**Figure 2-91. FP\_COMP2 Register**

31	30	29	28	27	26	25	24
REPLACE	RESERVED				COMP		
R/W-X	R/W-X				R/W-X		
23	22	21	20	19	18	17	16
			COMP				
			R/W-X				
15	14	13	12	11	10	9	8
			COMP				
			R/W-X				
7	6	5	4	3	2	1	0
			COMP		RESERVED	ENABLE	
			R/W-X		R/W-X	R/W-0h	

**Table 2-102. FP\_COMP2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	REPLACE	R/W	X	This selects what happens when the COMP address is matched. Settings other than b00 are only valid for instruction comparators. Literal comparators ignore non-b00 settings. Address remapping only takes place for the b00 setting. 0b (R/W) = remap to remap address. See FP_REMAP 1b (R/W) = set BKPT on lower halfword, upper is unaffected 10b (R/W) = set BKPT on upper halfword, lower is unaffected 11b (R/W) = set BKPT on both lower and upper halfwords.
29	RESERVED	R/W	X	
28-2	COMP	R/W	X	Comparison address.
1	RESERVED	R/W	X	
0	ENABLE	R/W	0h	Compare and remap enable for Flash Patch Comparator Register 2. The ENABLE bit of FP_CTRL must also be set to enable comparisons. Reset clears the ENABLE bit. 0b (R/W) = Flash Patch Comparator Register 2 compare and remap disabled 1b (R/W) = Flash Patch Comparator Register 2 compare and remap enabled

### 2.5.1.6 FP\_COMP3 Register (Offset = 14h) [reset = 00000000h]

Register mask: 00000001h

FP\_COMP3 is shown in [Figure 2-92](#) and described in [Table 2-103](#).

Flash Patch Comparator Registers. Use the Flash Patch Comparator Registers to store the values to compare with the PC address.

**Figure 2-92. FP\_COMP3 Register**

31	30	29	28	27	26	25	24
REPLACE	RESERVED				COMP		
R/W-X	R/W-X				R/W-X		
23	22	21	20	19	18	17	16
			COMP				
			R/W-X				
15	14	13	12	11	10	9	8
			COMP				
			R/W-X				
7	6	5	4	3	2	1	0
			COMP		RESERVED	ENABLE	
			R/W-X		R/W-X	R/W-0h	

**Table 2-103. FP\_COMP3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	REPLACE	R/W	X	This selects what happens when the COMP address is matched. Settings other than b00 are only valid for instruction comparators. Literal comparators ignore non-b00 settings. Address remapping only takes place for the b00 setting. 0b (R/W) = remap to remap address. See FP_REMAP 1b (R/W) = set BKPT on lower halfword, upper is unaffected 10b (R/W) = set BKPT on upper halfword, lower is unaffected 11b (R/W) = set BKPT on both lower and upper halfwords.
29	RESERVED	R/W	X	
28-2	COMP	R/W	X	Comparison address.
1	RESERVED	R/W	X	
0	ENABLE	R/W	0h	Compare and remap enable for Flash Patch Comparator Register 3. The ENABLE bit of FP_CTRL must also be set to enable comparisons. Reset clears the ENABLE bit. 0b (R/W) = Flash Patch Comparator Register 3 compare and remap disabled 1b (R/W) = Flash Patch Comparator Register 3 compare and remap enabled

### 2.5.1.7 FP\_COMP4 Register (Offset = 18h) [reset = 00000000h]

Register mask: 00000001h

FP\_COMP4 is shown in [Figure 2-93](#) and described in [Table 2-104](#).

Flash Patch Comparator Registers. Use the Flash Patch Comparator Registers to store the values to compare with the PC address.

**Figure 2-93. FP\_COMP4 Register**

31	30	29	28	27	26	25	24
REPLACE	RESERVED				COMP		
R/W-X	R/W-X				R/W-X		
23	22	21	20	19	18	17	16
			COMP				
			R/W-X				
15	14	13	12	11	10	9	8
			COMP				
			R/W-X				
7	6	5	4	3	2	1	0
			COMP		RESERVED	ENABLE	
			R/W-X		R/W-X	R/W-0h	

**Table 2-104. FP\_COMP4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	REPLACE	R/W	X	This selects what happens when the COMP address is matched. Settings other than b00 are only valid for instruction comparators. Literal comparators ignore non-b00 settings. Address remapping only takes place for the b00 setting. 0b (R/W) = remap to remap address. See FP_REMAP 1b (R/W) = set BKPT on lower halfword, upper is unaffected 10b (R/W) = set BKPT on upper halfword, lower is unaffected 11b (R/W) = set BKPT on both lower and upper halfwords.
29	RESERVED	R/W	X	
28-2	COMP	R/W	X	Comparison address.
1	RESERVED	R/W	X	
0	ENABLE	R/W	0h	Compare and remap enable for Flash Patch Comparator Register 4. The ENABLE bit of FP_CTRL must also be set to enable comparisons. Reset clears the ENABLE bit. 0b (R/W) = Flash Patch Comparator Register 4 compare and remap disabled 1b (R/W) = Flash Patch Comparator Register 4 compare and remap enabled

### 2.5.1.8 FP\_COMP5 Register (Offset = 1Ch) [reset = 00000000h]

Register mask: 00000001h

FP\_COMP5 is shown in [Figure 2-94](#) and described in [Table 2-105](#).

Flash Patch Comparator Registers. Use the Flash Patch Comparator Registers to store the values to compare with the PC address.

**Figure 2-94. FP\_COMP5 Register**

31	30	29	28	27	26	25	24
REPLACE		RESERVED			COMP		
R/W-X		R/W-X			R/W-X		
23	22	21	20	19	18	17	16
			COMP				
			R/W-X				
15	14	13	12	11	10	9	8
			COMP				
			R/W-X				
7	6	5	4	3	2	1	0
			COMP		RESERVED	ENABLE	
			R/W-X		R/W-X	R/W-0h	

**Table 2-105. FP\_COMP5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	REPLACE	R/W	X	This selects what happens when the COMP address is matched. Settings other than b00 are only valid for instruction comparators. Literal comparators ignore non-b00 settings. Address remapping only takes place for the b00 setting. 0b (R/W) = remap to remap address. See FP_REMAP 1b (R/W) = set BKPT on lower halfword, upper is unaffected 10b (R/W) = set BKPT on upper halfword, lower is unaffected 11b (R/W) = set BKPT on both lower and upper halfwords.
29	RESERVED	R/W	X	
28-2	COMP	R/W	X	Comparison address.
1	RESERVED	R/W	X	
0	ENABLE	R/W	0h	Compare and remap enable for Flash Patch Comparator Register 5. The ENABLE bit of FP_CTRL must also be set to enable comparisons. Reset clears the ENABLE bit. 0b (R/W) = Flash Patch Comparator Register 5 compare and remap disabled 1b (R/W) = Flash Patch Comparator Register 5 compare and remap enabled

### 2.5.1.9 FP\_COMP6 Register (Offset = 20h) [reset = 00000000h]

Register mask: 00000001h

FP\_COMP6 is shown in [Figure 2-95](#) and described in [Table 2-106](#).

Flash Patch Comparator Registers. Use the Flash Patch Comparator Registers to store the values to compare with the PC address.

**Figure 2-95. FP\_COMP6 Register**

31	30	29	28	27	26	25	24
REPLACE	RESERVED				COMP		
R/W-X	R/W-X				R/W-X		
23	22	21	20	19	18	17	16
			COMP				
			R/W-X				
15	14	13	12	11	10	9	8
			COMP				
			R/W-X				
7	6	5	4	3	2	1	0
			COMP		RESERVED	ENABLE	
			R/W-X		R/W-X	R/W-0h	

**Table 2-106. FP\_COMP6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	REPLACE	R/W	X	This selects what happens when the COMP address is matched. Settings other than b00 are only valid for instruction comparators. Literal comparators ignore non-b00 settings. Address remapping only takes place for the b00 setting. 0b (R/W) = remap to remap address. See FP_REMAP 1b (R/W) = set BKPT on lower halfword, upper is unaffected 10b (R/W) = set BKPT on upper halfword, lower is unaffected 11b (R/W) = set BKPT on both lower and upper halfwords.
29	RESERVED	R/W	X	
28-2	COMP	R/W	X	Comparison address.
1	RESERVED	R/W	X	
0	ENABLE	R/W	0h	Compare and remap enable for Flash Patch Comparator Register 6. The ENABLE bit of FP_CTRL must also be set to enable comparisons. Reset clears the ENABLE bit. 0b (R/W) = Flash Patch Comparator Register 6 compare and remap disabled 1b (R/W) = Flash Patch Comparator Register 6 compare and remap enabled

### 2.5.1.10 FP\_COMP7 Register (Offset = 24h) [reset = 00000000h]

Register mask: 00000001h

FP\_COMP7 is shown in [Figure 2-96](#) and described in [Table 2-107](#).

Flash Patch Comparator Registers. Use the Flash Patch Comparator Registers to store the values to compare with the PC address.

**Figure 2-96. FP\_COMP7 Register**

31	30	29	28	27	26	25	24
REPLACE	RESERVED				COMP		
R/W-X	R/W-X				R/W-X		
23	22	21	20	19	18	17	16
			COMP				
			R/W-X				
15	14	13	12	11	10	9	8
			COMP				
			R/W-X				
7	6	5	4	3	2	1	0
			COMP		RESERVED	ENABLE	
			R/W-X		R/W-X	R/W-0h	

**Table 2-107. FP\_COMP7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-30	REPLACE	R/W	X	This selects what happens when the COMP address is matched. Settings other than b00 are only valid for instruction comparators. Literal comparators ignore non-b00 settings. Address remapping only takes place for the b00 setting. 0b (R/W) = remap to remap address. See FP_REMAP 1b (R/W) = set BKPT on lower halfword, upper is unaffected 10b (R/W) = set BKPT on upper halfword, lower is unaffected 11b (R/W) = set BKPT on both lower and upper halfwords.
29	RESERVED	R/W	X	
28-2	COMP	R/W	X	Comparison address.
1	RESERVED	R/W	X	
0	ENABLE	R/W	0h	Compare and remap enable for Flash Patch Comparator Register 7. The ENABLE bit of FP_CTRL must also be set to enable comparisons. Reset clears the ENABLE bit. 0b (R/W) = Flash Patch Comparator Register 7 compare and remap disabled 1b (R/W) = Flash Patch Comparator Register 7 compare and remap enabled

## 2.5.2 DWT Registers

Table 2-108 lists the memory-mapped registers for the DWT. All register offset addresses not listed in Table 2-108 should be considered as reserved locations and the register contents should not be modified.

**Table 2-108. DWT Registers**

Offset	Acronym	Register Name	Type	Reset	Section
0h	CTRL	DWT Control Register	read-write	40000000h	Section 2.5.2.1
4h	CYCCNT	DWT Current PC Sampler Cycle Count Register	read-write	00000000h	Section 2.5.2.2
8h	CPICNT	DWT CPI Count Register	read-write	Undefined	Section 2.5.2.3
Ch	EXCCNT	DWT Exception Overhead Count Register	read-write	Undefined	Section 2.5.2.4
10h	SLEEPCNT	DWT Sleep Count Register	read-write	Undefined	Section 2.5.2.5
14h	LSUCNT	DWT LSU Count Register	read-write	Undefined	Section 2.5.2.6
18h	FOLDCNT	DWT Fold Count Register	read-write	Undefined	Section 2.5.2.7
1Ch	PCSR	DWT Program Counter Sample Register	read-only	Undefined	Section 2.5.2.8
20h	COMP0	DWT Comparator Register 0	read-write	Undefined	Section 2.5.2.9
24h	MASK0	DWT Mask Register 0	read-write	Undefined	Section 2.5.2.10
28h	FUNCTION0	DWT Function Register 0	read-write	00000000h	Section 2.5.2.11
30h	COMP1	DWT Comparator Register 1	read-write	Undefined	Section 2.5.2.12
34h	MASK1	DWT Mask Register 1	read-write	Undefined	Section 2.5.2.13
38h	FUNCTION1	DWT Function Register 1	read-write	00000200h	Section 2.5.2.14
40h	COMP2	DWT Comparator Register 2	read-write	Undefined	Section 2.5.2.15
44h	MASK2	DWT Mask Register 2	read-write	Undefined	Section 2.5.2.16
48h	FUNCTION2	DWT Function Register 2	read-write	00000000h	Section 2.5.2.17
50h	COMP3	DWT Comparator Register 3	read-write	Undefined	Section 2.5.2.18
54h	MASK3	DWT Mask Register 3	read-write	Undefined	Section 2.5.2.19
58h	FUNCTION3	DWT Function Register 3	read-write	00000000h	Section 2.5.2.20

### 2.5.2.1 CTRL Register (Offset = 0h) [reset = 40000000h]

CTRL is shown in [Figure 2-97](#) and described in [Table 2-109](#).

DWT Control Register. Use the DWT Control Register to enable the DWT unit.

**Figure 2-97. CTRL Register**

31	30	29	28	27	26	25	24
RESERVED						NOCYCCNT	NOPRFCNT
R/W-10h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTEN A	EXCEVTENA	CPIEVTE NA	EXCTRCENA
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED			PCSAMPLEEN A	SYNCTAP		CYCTAP	POSTCNT
R/W-0h			R/W-0h	R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
POSTCNT			POSTPRESET			CYCCNTENA	
R/W-0h			R/W-0h			R/W-0h	

**Table 2-109. CTRL Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-26	RESERVED	R/W	10h	
25	NOCYCCNT	R/W	0h	When set, DWT_CYCCNT is not supported.
24	NOPRFCNT	R/W	0h	When set, DWT_FOLDCNT, DWT_LSUCNT, DWT_SLEEPSCNT, DWT_EXCCNT, and DWT_CPICNT are not supported.
23	RESERVED	R/W	0h	
22	CYCEVTENA	R/W	0h	Enables Cycle count event. Emits an event when the POSTCNT counter triggers it. See CYCTAP (bit [9]) and POSTPRESET, bits [4:1], for details. This event is only emitted if PCSAMPLENA, bit [12], is disabled. PCSAMPLENA overrides the setting of this bit. Reset clears the CYCEVTENA bit. 0b (R/W) = Cycle count events disabled. 1b (R/W) = Cycle count events enabled.
21	FOLDEVTENA	R/W	0h	Enables Folded instruction count event. Emits an event when DWT_FOLDCNT overflows (every 256 cycles of folded instructions). A folded instruction is one that does not incur even one cycle to execute. For example, an IT instruction is folded away and so does not use up one cycle. Reset clears the FOLDEVTENA bit. 0b (R/W) = Folded instruction count events disabled. 1b (R/W) = Folded instruction count events enabled.
20	LSUEVTENA	R/W	0h	Enables LSU count event. Emits an event when DWT_LSUCNT overflows (every 256 cycles of LSU operation). LSU counts include all LSU costs after the initial cycle for the instruction. Reset clears the LSUEVTENA bit. 0b (R/W) = LSU count events disabled. 1b (R/W) = LSU count events enabled.
19	SLEEPEVTENA	R/W	0h	Enables Sleep count event. Emits an event when DWT_SLEEPSCNT overflows (every 256 cycles that the processor is sleeping). Reset clears the SLEEPEVTENA bit. 0b (R/W) = Sleep count events disabled. 1b (R/W) = Sleep count events enabled.

**Table 2-109. CTRL Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
18	EXCEVTENA	R/W	0h	Enables Interrupt overhead event. Emits an event when DWT_EXCCNT overflows (every 256 cycles of interrupt overhead). Reset clears the EXCEVTENA bit. 0b (R/W) = Interrupt overhead event disabled. 1b (R/W) = Interrupt overhead event enabled.
17	CPIEVTEA	R/W	0h	Enables CPI count event. Emits an event when DWT_CPICNT overflows (every 256 cycles of multi-cycle instructions). Reset clears the CPIEVTEA bit. 0b (R/W) = CPI counter events disabled. 1b (R/W) = CPI counter events enabled.
16	EXCTRCENA	R/W	0h	Enables Interrupt event tracing. Reset clears the EXCEVTENA bit. 0b (R/W) = interrupt event trace disabled. 1b (R/W) = interrupt event trace enabled.
15-13	RESERVED	R/W	0h	
12	PCSAMPLEENA	R/W	0h	Enables PC Sampling event. A PC sample event is emitted when the POSTCNT counter triggers it. See CYCTAP, bit [9], and POSTPRESET, bits [4:1], for details. Enabling this bit overrides CYCEVTENA (bit [20]). Reset clears the PCSAMPLENA bit. 0b (R/W) = PC Sampling event disabled. 1b (R/W) = Sampling event enabled.
11-10	SYNCTAP	R/W	0h	Feeds a synchronization pulse to the ITM SYNCENA control. The value selected here picks the rate (approximately 1/second or less) by selecting a tap on the DWT_CYCCNT register. To use synchronization (heartbeat and hot-connect synchronization), CYCCNTENA must be set to 1, SYNCTAP must be set to one of its values, and SYNCENA must be set to 1. 0b (R/W) = Disabled. No sync counting. 1b (R/W) = Tap at CYCCNT bit 24. 10b (R/W) = Tap at CYCCNT bit 26. 11b (R/W) = Tap at CYCCNT bit 28.
9	CYCTAP	R/W	0h	Selects a tap on the DWT_CYCCNT register. These are spaced at bits [6] and [10]. When the selected bit in the CYCCNT register changes from 0 to 1 or 1 to 0, it emits into the POSTCNT, bits [8:5], post-scalar counter. That counter then counts down. On a bit change when post-scalar is 0, it triggers an event for PC sampling or CYCEVTCNT. 0b (R/W) = selects bit [6] to tap 1b (R/W) = selects bit [10] to tap.
8-5	POSTCNT	R/W	0h	Post-scalar counter for CYCTAP. When the selected tapped bit changes from 0 to 1 or 1 to 0, the post scalar counter is down-counted when not 0. If 0, it triggers an event for PCSAMPLENA or CYCEVTENA use. It also reloads with the value from POSTPRESET (bits [4:1]).
4-1	POSTPRESET	R/W	0h	Reload value for POSTCNT, bits [8:5], post-scalar counter. If this value is 0, events are triggered on each tap change (a power of 2). If this field has a non-0 value, this forms a count-down value, to be reloaded into POSTCNT each time it reaches 0. For example, a value 1 in this register means an event is formed every other tap change.
0	CYCCNTENA	R/W	0h	Enable the CYCCNT counter. If not enabled, the counter does not count and no event is generated for PS sampling or CYCCNTENA. In normal use, the debugger must initialize the CYCCNT counter to 0.

### 2.5.2.2 CYCCNT Register (Offset = 4h) [reset = 00000000h]

CYCCNT is shown in [Figure 2-98](#) and described in [Table 2-110](#).

DWT Current PC Sampler Cycle Count Register. Use the DWT Current PC Sampler Cycle Count Register to count the number of core cycles. This count can measure elapsed execution time. This is a free-running counter. The counter has three functions: (1) When PCSAMPLENA is set, the PC is sampled and emitted when the selected tapped bit changes value (0 to 1 or 1 to 0) and any post-scalar value counts to 0. (2) When CYCEVTENA is set (and PCSAMPLENA is clear), an event is emitted when the selected tapped bit changes value (0 to 1 or 1 to 0) and any post-scalar value counts to 0. (3) Applications and debuggers can use the counter to measure elapsed execution time. By subtracting a start and an end time, an application can measure time between in-core clocks (other than when Halted in debug). This is valid to  $2^{32}$  core clock cycles (for example, almost 86 seconds at 50MHz).

**Figure 2-98. CYCCNT Register**

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
CYCCNT																															
R/W-0h																															

**Table 2-110. CYCCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CYCCNT	R/W	0h	Current PC Sampler Cycle Counter count value. When enabled, this counter counts the number of core cycles, except when the core is halted. CYCCNT is a free running counter, counting upwards. It wraps around to 0 on overflow. The debugger must initialize this to 0 when first enabling.

### 2.5.2.3 CPICNT Register (Offset = 8h) [reset = Undefined]

CPICNT is shown in [Figure 2-99](#) and described in [Table 2-111](#).

DWT CPI Count Register. Use the DWT CPI Count Register to count the total number of instruction cycles beyond the first cycle.

**Figure 2-99. CPICNT Register**

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
RESERVED																										CPICNT					
R/W																										R/W					

**Table 2-111. CPICNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R/W	Undefined	
7-0	CPICNT	R/W	Undefined	Current CPI counter value. Increments on the additional cycles (the first cycle is not counted) required to execute all instructions except those recorded by DWT_LSUCNT. This counter also increments on all instruction fetch stalls. If CPIEVTEVA is set, an event is emitted when the counter overflows. Clears to 0 on enabling.

#### 2.5.2.4 EXCCNT Register (Offset = Ch) [reset = Undefined]

EXCCNT is shown in [Figure 2-100](#) and described in [Table 2-112](#).

DWT Exception Overhead Count Register. Use the DWT Exception Overhead Count Register to count the total cycles spent in interrupt processing.

**Figure 2-100. EXCCNT Register**

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
RESERVED																										EXCCNT					
R/W																										R/W					

**Table 2-112. EXCCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R/W	Undefined	
7-0	EXCCNT	R/W	Undefined	Current interrupt overhead counter value. Counts the total cycles spent in interrupt processing (for example entry stacking, return unstacking, pre-emption). An event is emitted on counter overflow (every 256 cycles). This counter initializes to 0 when enabled. Clears to 0 on enabling.

### 2.5.2.5 SLEEPCNT Register (Offset = 10h) [reset = Undefined]

SLEEPCNT is shown in [Figure 2-101](#) and described in [Table 2-113](#).

DWT Sleep Count Register. Use the DWT Sleep Count Register to count the total number of cycles during which the processor is sleeping.

**Figure 2-101. SLEEPCNT Register**

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
RESERVED																								SLEEPCNT							
R/W																								R/W							

**Table 2-113. SLEEPCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R/W	Undefined	
7-0	SLEEPCNT	R/W	Undefined	Sleep counter. Counts the number of cycles during which the processor is sleeping. An event is emitted on counter overflow (every 256 cycles). This counter initializes to 0 when enabled. Note that SLEEPCNT is clocked using FCLK. It is possible that the frequency of FCLK might be reduced while the processor is sleeping to minimize power consumption. This means that sleep duration must be calculated with the frequency of FCLK during sleep.

### 2.5.2.6 LSUCNT Register (Offset = 14h) [reset = Undefined]

LSUCNT is shown in [Figure 2-102](#) and described in [Table 2-114](#).

DWT LSU Count Register. Use the DWT LSU Count Register to count the total number of cycles during which the processor is processing an LSU operation beyond the first cycle.

**Figure 2-102. LSUCNT Register**

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
RESERVED																									LSUCNT						
R/W																									R/W						

**Table 2-114. LSUCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R/W	Undefined	
7-0	LSUCNT	R/W	Undefined	LSU counter. This counts the total number of cycles that the processor is processing an LSU operation. The initial execution cost of the instruction is not counted. For example, an LDR that takes two cycles to complete increments this counter one cycle. Equivalently, an LDR that stalls for two cycles (and so takes four cycles), increments this counter three times. An event is emitted on counter overflow (every 256 cycles). Clears to 0 on enabling.

### 2.5.2.7 FOLDCNT Register (Offset = 18h) [reset = Undefined]

FOLDCNT is shown in [Figure 2-103](#) and described in [Table 2-115](#).

DWT Fold Count Register. Use the DWT Fold Count Register to count the total number of folded instructions. This counts 1 for each instruction that takes 0 cycles.

**Figure 2-103. FOLDCNT Register**

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
RESERVED																									FOLDCNT						
R/W																									R/W						

**Table 2-115. FOLDCNT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R/W	Undefined	
7-0	FOLDCNT	R/W	Undefined	This counts the total number folded instructions. This counter initializes to 0 when enabled.

### 2.5.2.8 PCSR Register (Offset = 1Ch) [reset = Undefined]

PCSR is shown in [Figure 2-104](#) and described in [Table 2-116](#).

DWT Program Counter Sample Register. Use the DWT Program Counter Sample Register (PCSR) to enable coarse-grained software profiling using a debug agent, without changing the currently executing code. If the core is not in debug state, the value returned is the instruction address of a recently executed instruction. If the core is in debug state, the value returned is 0xFFFFFFFF.

**Figure 2-104. PCSR Register**

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
EIASAMPLE																															
R																															

**Table 2-116. PCSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	EIASAMPLE	R	Undefined	Execution instruction address sample, or 0xFFFFFFFF if the core is halted.

### 2.5.2.9 COMP0 Register (Offset = 20h) [reset = Undefined]

COMP0 is shown in [Figure 2-105](#) and described in [Table 2-117](#).

DWT Comparator Register 0. Use the DWT Comparator Registers 0-3 to write the values that trigger watchpoint events.

**Figure 2-105. COMP0 Register**

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
COMP																															
R/W																															

**Table 2-117. COMP0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	COMP	R/W	Undefined	Data value to compare against PC and the data address as given by DWT_FUNCTION0. DWT_COMP0 can also compare against the value of the PC Sampler Counter (DWT_CYCCNT).

### 2.5.2.10 MASK0 Register (Offset = 24h) [reset = Undefined]

MASK0 is shown in [Figure 2-106](#) and described in [Table 2-118](#).

DWT Mask Register 0. Use the DWT Mask Registers 0-3 to apply a mask to data addresses when matching against COMP.

**Figure 2-106. MASK0 Register**

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
RESERVED																										MASK					
R/W																										R/W					

**Table 2-118. MASK0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R/W	Undefined	
3-0	MASK	R/W	Undefined	Mask on data address when matching against COMP. This is the size of the ignore mask. That is, DWT matching is performed as:(ADDR ANDed with (~0 left bit-shifted by MASK)) == COMP. However, the actual comparison is slightly more complex to enable matching an address wherever it appears on a bus. So, if COMP is 3, this matches a word access of 0, because 3 would be within the word.

### 2.5.2.11 FUNCTION0 Register (Offset = 28h) [reset = 00000000h]

FUNCTION0 is shown in [Figure 2-107](#) and described in [Table 2-119](#).

DWT Function Register 0. Use the DWT Function Registers 0-3 to control the operation of the comparator. Each comparator can: 1. Match against either the PC or the data address. This is controlled by CYCMATCH. This function is only available for comparator 0 (DWT\_COMP0). 2. Perform data value comparisons if associated address comparators have performed an address match. This function is only available for comparator 1 (DWT\_COMP1). 3. Emit data or PC couples, trigger the ETM, or generate a watchpoint depending on the operation defined by FUNCTION.

**Figure 2-107. FUNCTION0 Register**

31	30	29	28	27	26	25	24
RESERVED							MATCHED
R/W-0h							R/W-0h
23	22	21	20	19	18	17	16
RESERVED				DATAVADDR1			
R/W-0h							R/W-0h
15	14	13	12	11	10	9	8
DATAVADDR0				DATAVSIZE	LNK1ENA	DATAVMATCH	
R/W-0h							R-0h R/W-0h
7	6	5	4	3	2	1	0
RESERVED	EMITRANGE	RESERVED	FUNCTION				
R/W-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 2-119. FUNCTION0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R/W	0h	
24	MATCHED	R/W	0h	This bit is set when the comparator matches, and indicates that the operation defined by FUNCTION has occurred since this bit was last read. This bit is cleared on read.
23-20	RESERVED	R/W	0h	
19-16	DATAVADDR1	R/W	0h	Identity of a second linked address comparator for data value matching when DATAVMATCH == 1 and LNK1ENA == 1.
15-12	DATAVADDR0	R/W	0h	Identity of a linked address comparator for data value matching when DATAVMATCH == 1.
11-10	DATAVSIZE	R/W	0h	Defines the size of the data in the COMP register that is to be matched: 0b (R/W) = byte 1b (R/W) = halfword 10b (R/W) = word 11b (R/W) = Unpredictable.
9	LNK1ENA	R	0h	
8	DATAVMATCH	R/W	0h	This bit is only available in comparator 1. When DATAVMATCH is set, this comparator performs data value compares. The comparators given by DATAVADDR0 and DATAVADDR1 provide the address for the data comparison. If DATAVMATCH is set in DWT_FUNCTION1, the FUNCTION setting for the comparators given by DATAVADDR0 and DATAVADDR1 are overridden and those comparators only provide the address match for the data comparison.
7-6	RESERVED	R/W	0h	
5	EMITRANGE	R/W	0h	Emit range field. Reserved to permit emitting offset when range match occurs. Reset clears the EMITRANGE bit. PC sampling is not supported when EMITRANGE is enabled. EMITRANGE only applies for: FUNCTION = b0001, b0010, b0011, b1100, b1101, b1110, and b1111.

**Table 2-119. FUNCTION0 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	RESERVED	R/W	0h	
3-0	FUNCTION	R/W	0h	<p>Function settings. Note 1: If the ETM is not fitted, then ETM trigger is not possible. Note 2: Data value is only sampled for accesses that do not fault (MPU or bus fault). The PC is sampled irrespective of any faults. The PC is only sampled for the first address of a burst. Note 3: PC match is not recommended for watchpoints because it stops after the instruction. It mainly guards and triggers the ETM.</p> <p>0b (R/W) = Disabled      1b (R/W) = EMITRANGE = 0, sample and emit PC through ITM.      EMITRANGE = 1, emit address offset through ITM      10b (R/W) = EMITRANGE = 0, emit data through ITM on read and write. EMITRANGE = 1, emit data and address offset through ITM on read or write.      11b (R/W) = EMITRANGE = 0, sample PC and data value through ITM on read or write. EMITRANGE = 1, emit address offset and data value through ITM on read or write.      100b (R/W) = Watchpoint on PC match.      101b (R/W) = Watchpoint on read.      110b (R/W) = Watchpoint on write.      111b (R/W) = Watchpoint on read or write.      1000b (R/W) = ETM trigger on PC match      1001b (R/W) = ETM trigger on read      1010b (R/W) = ETM trigger on write      1011b (R/W) = ETM trigger on read or write      1100b (R/W) = EMITRANGE = 0, sample data for read transfers.      EMITRANGE = 1, sample Daddr [15:0] for read transfers      1101b (R/W) = EMITRANGE = 0, sample data for write transfers.      EMITRANGE = 1, sample Daddr [15:0] for write transfers      1110b (R/W) = EMITRANGE = 0, sample PC + data for read transfers. EMITRANGE = 1, sample Daddr [15:0] + data for read transfers      1111b (R/W) = EMITRANGE = 0, sample PC + data for write transfers. EMITRANGE = 1, sample Daddr [15:0] + data for write transfers</p>

### **2.5.2.12 COMP1 Register (Offset = 30h) [reset = Undefined]**

COMP1 is shown in [Figure 2-108](#) and described in [Table 2-120](#).

DWT Comparator Register 1. Use the DWT Comparator Registers 0-3 to write the values that trigger watchpoint events.

**Figure 2-108. COMP1 Register**

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
COMP																															
R/W																															

**Table 2-120. COMP1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	COMP	R/W	Undefined	Data value to compare against PC and the data address as given by DWT_FUNCTION1.

### 2.5.2.13 MASK1 Register (Offset = 34h) [reset = Undefined]

MASK1 is shown in [Figure 2-109](#) and described in [Table 2-121](#).

DWT Mask Register 1. Use the DWT Mask Registers 0-3 to apply a mask to data addresses when matching against COMP.

**Figure 2-109. MASK1 Register**

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
RESERVED																										MASK					
R/W																										R/W					

**Table 2-121. MASK1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R/W	Undefined	
3-0	MASK	R/W	Undefined	Mask on data address when matching against COMP. This is the size of the ignore mask. That is, DWT matching is performed as:(ADDR ANDed with (~0 left bit-shifted by MASK)) == COMP. However, the actual comparison is slightly more complex to enable matching an address wherever it appears on a bus. So, if COMP is 3, this matches a word access of 0, because 3 would be within the word.

### 2.5.2.14 FUNCTION1 Register (Offset = 38h) [reset = 00000200h]

FUNCTION1 is shown in [Figure 2-110](#) and described in [Table 2-122](#).

DWT Function Register 1. Use the DWT Function Registers 0-3 to control the operation of the comparator. Each comparator can: 1. Match against either the PC or the data address. This is controlled by CYCMATCH. This function is only available for comparator 0 (DWT\_COMP0). 2. Perform data value comparisons if associated address comparators have performed an address match. This function is only available for comparator 1 (DWT\_COMP1). 3. Emit data or PC couples, trigger the ETM, or generate a watchpoint depending on the operation defined by FUNCTION.

**Figure 2-110. FUNCTION1 Register**

31	30	29	28	27	26	25	24
RESERVED							MATCHED
R/W-0h							R/W-0h
23	22	21	20	19	18	17	16
RESERVED				DATAVADDR1			
R/W-0h							R/W-0h
15	14	13	12	11	10	9	8
DATAVADDR0				DATAVSIZE	LNK1ENA	DATAVMATCH	
R/W-0h							R-1h R/W-0h
7	6	5	4	3	2	1	0
CYCMATCH	RESERVED	EMITRANGE	RESERVED	FUNCTION			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			

**Table 2-122. FUNCTION1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R/W	0h	
24	MATCHED	R/W	0h	This bit is set when the comparator matches, and indicates that the operation defined by FUNCTION has occurred since this bit was last read. This bit is cleared on read.
23-20	RESERVED	R/W	0h	
19-16	DATAVADDR1	R/W	0h	Identity of a second linked address comparator for data value matching when DATAVMATCH == 1 and LNK1ENA == 1.
15-12	DATAVADDR0	R/W	0h	Identity of a linked address comparator for data value matching when DATAVMATCH == 1.
11-10	DATAVSIZE	R/W	0h	Defines the size of the data in the COMP register that is to be matched: 0b (R/W) = byte 1b (R/W) = halfword 10b (R/W) = word 11b (R/W) = Unpredictable.
9	LNK1ENA	R	1h	
8	DATAVMATCH	R/W	0h	This bit is only available in comparator 1. When DATAVMATCH is set, this comparator performs data value compares. The comparators given by DATAVADDR0 and DATAVADDR1 provide the address for the data comparison. If DATAVMATCH is set in DWT_FUNCTION1, the FUNCTION setting for the comparators given by DATAVADDR0 and DATAVADDR1 are overridden and those comparators only provide the address match for the data comparison.
7	CYCMATCH	R/W	0h	Only available in comparator 0. When set, this comparator compares against the clock cycle counter.
6	RESERVED	R/W	0h	

**Table 2-122. FUNCTION1 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
5	EMITRANGE	R/W	0h	Emit range field. Reserved to permit emitting offset when range match occurs. Reset clears the EMITRANGE bit. PC sampling is not supported when EMITRANGE is enabled. EMITRANGE only applies for: FUNCTION = b0001, b0010, b0011, b1100, b1101, b1110, and b1111.
4	RESERVED	R/W	0h	
3-0	FUNCTION	R/W	0h	<p>Function settings. Note 1: If the ETM is not fitted, then ETM trigger is not possible. Note 2: Data value is only sampled for accesses that do not fault (MPU or bus fault). The PC is sampled irrespective of any faults. The PC is only sampled for the first address of a burst. Note 3: FUNCTION is overridden for comparators given by DATAVADDR0 and DATAVADDR1 in DWT_FUNCTION1if DATAVMATCH is also set in DWT_FUNCTION1. The comparators given by DATAVADDR0 and DATAVADDR1 can then only perform address comparator matches for comparator 1 data matches. Note 4: If the data matching functionality is not included during implementation it is not possible to set DATAVADDR0, DATAVADDR1, or DATAVMATCH in DWT_FUNCTION1. This means that the data matching functionality is not available in the implementation. Test the availability of data matching by writing and reading the DATAVMATCH bit in DWT_FUNCTION1. If it is not settable then data matching is unavailable. Note 5: PC match is not recommended for watchpoints because it stops after the instruction. It mainly guards and triggers the ETM.</p> <p>0b (R/W) = Disabled      1b (R/W) = EMITRANGE = 0, sample and emit PC through ITM.      EMITRANGE = 1, emit address offset through ITM      10b (R/W) = EMITRANGE = 0, emit data through ITM on read and write. EMITRANGE = 1, emit data and address offset through ITM on read or write.      11b (R/W) = EMITRANGE = 0, sample PC and data value through ITM on read or write. EMITRANGE = 1, emit address offset and data value through ITM on read or write.      100b (R/W) = Watchpoint on PC match.      101b (R/W) = Watchpoint on read.      110b (R/W) = Watchpoint on write.      111b (R/W) = Watchpoint on read or write.      1000b (R/W) = ETM trigger on PC match      1001b (R/W) = ETM trigger on read      1010b (R/W) = ETM trigger on write      1011b (R/W) = ETM trigger on read or write      1100b (R/W) = EMITRANGE = 0, sample data for read transfers. EMITRANGE = 1, sample Daddr [15:0] for read transfers      1101b (R/W) = EMITRANGE = 0, sample data for write transfers. EMITRANGE = 1, sample Daddr [15:0] for write transfers      1110b (R/W) = EMITRANGE = 0, sample PC + data for read transfers. EMITRANGE = 1, sample Daddr [15:0] + data for read transfers      1111b (R/W) = EMITRANGE = 0, sample PC + data for write transfers. EMITRANGE = 1, sample Daddr [15:0] + data for write transfers</p>

### **2.5.2.15 COMP2 Register (Offset = 40h) [reset = Undefined]**

COMP2 is shown in [Figure 2-111](#) and described in [Table 2-123](#).

DWT Comparator Register 2. Use the DWT Comparator Registers 0-3 to write the values that trigger watchpoint events.

**Figure 2-111. COMP2 Register**

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
COMP																															
R/W																															

**Table 2-123. COMP2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	COMP	R/W	Undefined	Data value to compare against PC and the data address as given by DWT_FUNCTION2.

### 2.5.2.16 MASK2 Register (Offset = 44h) [reset = Undefined]

MASK2 is shown in [Figure 2-112](#) and described in [Table 2-124](#).

DWT Mask Register 2. Use the DWT Mask Registers 0-3 to apply a mask to data addresses when matching against COMP.

**Figure 2-112. MASK2 Register**

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
RESERVED																										MASK					
R/W																										R/W					

**Table 2-124. MASK2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R/W	Undefined	
3-0	MASK	R/W	Undefined	Mask on data address when matching against COMP. This is the size of the ignore mask. That is, DWT matching is performed as:(ADDR ANDed with (~0 left bit-shifted by MASK)) == COMP. However, the actual comparison is slightly more complex to enable matching an address wherever it appears on a bus. So, if COMP is 3, this matches a word access of 0, because 3 would be within the word.

### 2.5.2.17 FUNCTION2 Register (Offset = 48h) [reset = 00000000h]

FUNCTION2 is shown in [Figure 2-113](#) and described in [Table 2-125](#).

DWT Function Register 2. Use the DWT Function Registers 0-3 to control the operation of the comparator. Each comparator can: 1. Match against either the PC or the data address. This is controlled by CYCMATCH. This function is only available for comparator 0 (DWT\_COMP0). 2. Perform data value comparisons if associated address comparators have performed an address match. This function is only available for comparator 1 (DWT\_COMP1). 3. Emit data or PC couples, trigger the ETM, or generate a watchpoint depending on the operation defined by FUNCTION.

**Figure 2-113. FUNCTION2 Register**

31	30	29	28	27	26	25	24
RESERVED							MATCHED
R/W-0h							R/W-0h
23	22	21	20	19	18	17	16
RESERVED				DATAVADDR1			
R/W-0h							R/W-0h
15	14	13	12	11	10	9	8
DATAVADDR0				DATAVSIZE	LNK1ENA	DATAVMATCH	
R/W-0h							R-0h R/W-0h
7	6	5	4	3	2	1	0
RESERVED	EMITRANGE	RESERVED	FUNCTION				
R/W-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 2-125. FUNCTION2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R/W	0h	
24	MATCHED	R/W	0h	This bit is set when the comparator matches, and indicates that the operation defined by FUNCTION has occurred since this bit was last read. This bit is cleared on read.
23-20	RESERVED	R/W	0h	
19-16	DATAVADDR1	R/W	0h	Identity of a second linked address comparator for data value matching when DATAVMATCH == 1 and LNK1ENA == 1.
15-12	DATAVADDR0	R/W	0h	Identity of a linked address comparator for data value matching when DATAVMATCH == 1.
11-10	DATAVSIZE	R/W	0h	Defines the size of the data in the COMP register that is to be matched: 0b (R/W) = byte 1b (R/W) = halfword 10b (R/W) = word 11b (R/W) = Unpredictable.
9	LNK1ENA	R	0h	
8	DATAVMATCH	R/W	0h	This bit is only available in comparator 1. When DATAVMATCH is set, this comparator performs data value compares. The comparators given by DATAVADDR0 and DATAVADDR1 provide the address for the data comparison. If DATAVMATCH is set in DWT_FUNCTION1, the FUNCTION setting for the comparators given by DATAVADDR0 and DATAVADDR1 are overridden and those comparators only provide the address match for the data comparison.
7-6	RESERVED	R/W	0h	
5	EMITRANGE	R/W	0h	Emit range field. Reserved to permit emitting offset when range match occurs. Reset clears the EMITRANGE bit. PC sampling is not supported when EMITRANGE is enabled. EMITRANGE only applies for: FUNCTION = b0001, b0010, b0011, b1100, b1101, b1110, and b1111.

**Table 2-125. FUNCTION2 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	RESERVED	R/W	0h	
3-0	FUNCTION	R/W	0h	<p>Function settings. Note 1: If the ETM is not fitted, then ETM trigger is not possible. Note 2: Data value is only sampled for accesses that do not fault (MPU or bus fault). The PC is sampled irrespective of any faults. The PC is only sampled for the first address of a burst. Note 3: PC match is not recommended for watchpoints because it stops after the instruction. It mainly guards and triggers the ETM.</p> <p>0b (R/W) = Disabled            1b (R/W) = EMITRANGE = 0, sample and emit PC through ITM.            EMITRANGE = 1, emit address offset through ITM            10b (R/W) = EMITRANGE = 0, emit data through ITM on read and write. EMITRANGE = 1, emit data and address offset through ITM on read or write.            11b (R/W) = EMITRANGE = 0, sample PC and data value through ITM on read or write. EMITRANGE = 1, emit address offset and data value through ITM on read or write.            100b (R/W) = Watchpoint on PC match.            101b (R/W) = Watchpoint on read.            110b (R/W) = Watchpoint on write.            111b (R/W) = Watchpoint on read or write.            1000b (R/W) = ETM trigger on PC match            1001b (R/W) = ETM trigger on read            1010b (R/W) = ETM trigger on write            1011b (R/W) = ETM trigger on read or write            1100b (R/W) = EMITRANGE = 0, sample data for read transfers.            EMITRANGE = 1, sample Daddr [15:0] for read transfers            1101b (R/W) = EMITRANGE = 0, sample data for write transfers.            EMITRANGE = 1, sample Daddr [15:0] for write transfers            1110b (R/W) = EMITRANGE = 0, sample PC + data for read transfers. EMITRANGE = 1, sample Daddr [15:0] + data for read transfers            1111b (R/W) = EMITRANGE = 0, sample PC + data for write transfers. EMITRANGE = 1, sample Daddr [15:0] + data for write transfers</p>

### **2.5.2.18 COMP3 Register (Offset = 50h) [reset = Undefined]**

COMP3 is shown in [Figure 2-114](#) and described in [Table 2-126](#).

DWT Comparator Register 3. Use the DWT Comparator Registers 0-3 to write the values that trigger watchpoint events.

**Figure 2-114. COMP3 Register**

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
COMP																															
R/W																															

**Table 2-126. COMP3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	COMP	R/W	Undefined	Data value to compare against PC and the data address as given by DWT_FUNCTION3.

### 2.5.2.19 MASK3 Register (Offset = 54h) [reset = Undefined]

MASK3 is shown in [Figure 2-115](#) and described in [Table 2-127](#).

DWT Mask Register 3. Use the DWT Mask Registers 0-3 to apply a mask to data addresses when matching against COMP.

**Figure 2-115. MASK3 Register**

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
RESERVED																													MASK		
R/W																													R/W		

**Table 2-127. MASK3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R/W	Undefined	
3-0	MASK	R/W	Undefined	Mask on data address when matching against COMP. This is the size of the ignore mask. That is, DWT matching is performed as:(ADDR ANDed with (~0 left bit-shifted by MASK)) == COMP. However, the actual comparison is slightly more complex to enable matching an address wherever it appears on a bus. So, if COMP is 3, this matches a word access of 0, because 3 would be within the word.

### 2.5.2.20 FUNCTION3 Register (Offset = 58h) [reset = 00000000h]

FUNCTION3 is shown in [Figure 2-116](#) and described in [Table 2-128](#).

DWT Function Register 3. Use the DWT Function Registers 0-3 to control the operation of the comparator. Each comparator can: 1. Match against either the PC or the data address. This is controlled by CYCMATCH. This function is only available for comparator 0 (DWT\_COMP0). 2. Perform data value comparisons if associated address comparators have performed an address match. This function is only available for comparator 1 (DWT\_COMP1). 3. Emit data or PC couples, trigger the ETM, or generate a watchpoint depending on the operation defined by FUNCTION.

**Figure 2-116. FUNCTION3 Register**

31	30	29	28	27	26	25	24
RESERVED							MATCHED
R/W-0h							R/W-0h
23	22	21	20	19	18	17	16
RESERVED				DATAVADDR1			
R/W-0h							R/W-0h
15	14	13	12	11	10	9	8
DATAVADDR0				DATAVSIZE	LNK1ENA	DATAVMATCH	
R/W-0h							R-0h R/W-0h
7	6	5	4	3	2	1	0
RESERVED	EMITRANGE	RESERVED	FUNCTION				
R/W-0h		R/W-0h	R/W-0h	R/W-0h			

**Table 2-128. FUNCTION3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-25	RESERVED	R/W	0h	
24	MATCHED	R/W	0h	This bit is set when the comparator matches, and indicates that the operation defined by FUNCTION has occurred since this bit was last read. This bit is cleared on read.
23-20	RESERVED	R/W	0h	
19-16	DATAVADDR1	R/W	0h	Identity of a second linked address comparator for data value matching when DATAVMATCH == 1 and LNK1ENA == 1.
15-12	DATAVADDR0	R/W	0h	Identity of a linked address comparator for data value matching when DATAVMATCH == 1.
11-10	DATAVSIZE	R/W	0h	Defines the size of the data in the COMP register that is to be matched: 0b (R/W) = byte 1b (R/W) = halfword 10b (R/W) = word 11b (R/W) = Unpredictable.
9	LNK1ENA	R	0h	
8	DATAVMATCH	R/W	0h	This bit is only available in comparator 1. When DATAVMATCH is set, this comparator performs data value compares. The comparators given by DATAVADDR0 and DATAVADDR1 provide the address for the data comparison. If DATAVMATCH is set in DWT_FUNCTION1, the FUNCTION setting for the comparators given by DATAVADDR0 and DATAVADDR1 are overridden and those comparators only provide the address match for the data comparison.
7-6	RESERVED	R/W	0h	
5	EMITRANGE	R/W	0h	Emit range field. Reserved to permit emitting offset when range match occurs. Reset clears the EMITRANGE bit. PC sampling is not supported when EMITRANGE is enabled. EMITRANGE only applies for: FUNCTION = b0001, b0010, b0011, b1100, b1101, b1110, and b1111.

**Table 2-128. FUNCTION3 Register Field Descriptions (continued)**

Bit	Field	Type	Reset	Description
4	RESERVED	R/W	0h	
3-0	FUNCTION	R/W	0h	<p>Function settings. Note 1: If the ETM is not fitted, then ETM trigger is not possible. Note 2: Data value is only sampled for accesses that do not fault (MPU or bus fault). The PC is sampled irrespective of any faults. The PC is only sampled for the first address of a burst. Note 3: PC match is not recommended for watchpoints because it stops after the instruction. It mainly guards and triggers the ETM.</p> <p>0b (R/W) = Disabled      1b (R/W) = EMITRANGE = 0, sample and emit PC through ITM.      EMITRANGE = 1, emit address offset through ITM      10b (R/W) = EMITRANGE = 0, emit data through ITM on read and write. EMITRANGE = 1, emit data and address offset through ITM on read or write.      11b (R/W) = EMITRANGE = 0, sample PC and data value through ITM on read or write. EMITRANGE = 1, emit address offset and data value through ITM on read or write.      100b (R/W) = Watchpoint on PC match.      101b (R/W) = Watchpoint on read.      110b (R/W) = Watchpoint on write.      111b (R/W) = Watchpoint on read or write.      1000b (R/W) = ETM trigger on PC match      1001b (R/W) = ETM trigger on read      1010b (R/W) = ETM trigger on write      1011b (R/W) = ETM trigger on read or write      1100b (R/W) = EMITRANGE = 0, sample data for read transfers.      EMITRANGE = 1, sample Daddr [15:0] for read transfers      1101b (R/W) = EMITRANGE = 0, sample data for write transfers.      EMITRANGE = 1, sample Daddr [15:0] for write transfers      1110b (R/W) = EMITRANGE = 0, sample PC + data for read transfers. EMITRANGE = 1, sample Daddr [15:0] + data for read transfers      1111b (R/W) = EMITRANGE = 0, sample PC + data for write transfers. EMITRANGE = 1, sample Daddr [15:0] + data for write transfers</p>

### 2.5.3 ITM Registers

Table 2-129 lists the memory-mapped registers for the ITM. All register offset addresses not listed in Table 2-129 should be considered as reserved locations and the register contents should not be modified.

**Table 2-129. ITM Registers**

Offset	Acronym	Register Name	Type	Reset	Section
0h	STIM0	ITM Stimulus Port 0	read-write	Undefined	<a href="#">Section 2.5.3.1</a>
4h	STIM1	ITM Stimulus Port 1	read-write	Undefined	<a href="#">Section 2.5.3.2</a>
8h	STIM2	ITM Stimulus Port 2	read-write	Undefined	<a href="#">Section 2.5.3.3</a>
Ch	STIM3	ITM Stimulus Port 3	read-write	Undefined	<a href="#">Section 2.5.3.4</a>
10h	STIM4	ITM Stimulus Port 4	read-write	Undefined	<a href="#">Section 2.5.3.5</a>
14h	STIM5	ITM Stimulus Port 5	read-write	Undefined	<a href="#">Section 2.5.3.6</a>
18h	STIM6	ITM Stimulus Port 6	read-write	Undefined	<a href="#">Section 2.5.3.7</a>
1Ch	STIM7	ITM Stimulus Port 7	read-write	Undefined	<a href="#">Section 2.5.3.8</a>
20h	STIM8	ITM Stimulus Port 8	read-write	Undefined	<a href="#">Section 2.5.3.9</a>
24h	STIM9	ITM Stimulus Port 9	read-write	Undefined	<a href="#">Section 2.5.3.10</a>
28h	STIM10	ITM Stimulus Port 10	read-write	Undefined	<a href="#">Section 2.5.3.11</a>
2Ch	STIM11	ITM Stimulus Port 11	read-write	Undefined	<a href="#">Section 2.5.3.12</a>
30h	STIM12	ITM Stimulus Port 12	read-write	Undefined	<a href="#">Section 2.5.3.13</a>
34h	STIM13	ITM Stimulus Port 13	read-write	Undefined	<a href="#">Section 2.5.3.14</a>
38h	STIM14	ITM Stimulus Port 14	read-write	Undefined	<a href="#">Section 2.5.3.15</a>
3Ch	STIM15	ITM Stimulus Port 15	read-write	Undefined	<a href="#">Section 2.5.3.16</a>
40h	STIM16	ITM Stimulus Port 16	read-write	Undefined	<a href="#">Section 2.5.3.17</a>
44h	STIM17	ITM Stimulus Port 17	read-write	Undefined	<a href="#">Section 2.5.3.18</a>
48h	STIM18	ITM Stimulus Port 18	read-write	Undefined	<a href="#">Section 2.5.3.19</a>
4Ch	STIM19	ITM Stimulus Port 19	read-write	Undefined	<a href="#">Section 2.5.3.20</a>
50h	STIM20	ITM Stimulus Port 20	read-write	Undefined	<a href="#">Section 2.5.3.21</a>
54h	STIM21	ITM Stimulus Port 21	read-write	Undefined	<a href="#">Section 2.5.3.22</a>
58h	STIM22	ITM Stimulus Port 22	read-write	Undefined	<a href="#">Section 2.5.3.23</a>
5Ch	STIM23	ITM Stimulus Port 23	read-write	Undefined	<a href="#">Section 2.5.3.24</a>
60h	STIM24	ITM Stimulus Port 24	read-write	Undefined	<a href="#">Section 2.5.3.25</a>
64h	STIM25	ITM Stimulus Port 25	read-write	Undefined	<a href="#">Section 2.5.3.26</a>
68h	STIM26	ITM Stimulus Port 26	read-write	Undefined	<a href="#">Section 2.5.3.27</a>
6Ch	STIM27	ITM Stimulus Port 27	read-write	Undefined	<a href="#">Section 2.5.3.28</a>
70h	STIM28	ITM Stimulus Port 28	read-write	Undefined	<a href="#">Section 2.5.3.29</a>
74h	STIM29	ITM Stimulus Port 29	read-write	Undefined	<a href="#">Section 2.5.3.30</a>
78h	STIM30	ITM Stimulus Port 30	read-write	Undefined	<a href="#">Section 2.5.3.31</a>
7Ch	STIM31	ITM Stimulus Port 31	read-write	Undefined	<a href="#">Section 2.5.3.32</a>
E00h	TER	ITM Trace Enable Register	read-write	00000000h	<a href="#">Section 2.5.3.33</a>
E40h	TPR	ITM Trace Privilege Register	read-write	00000000h	<a href="#">Section 2.5.3.34</a>
E80h	TCR	ITM Trace Control Register	read-write	00000000h	<a href="#">Section 2.5.3.35</a>
EF8h	IWR	ITM Integration Write Register	write-only	00000000h	<a href="#">Section 2.5.3.36</a>
F00h	IMCR	ITM Integration Mode Control Register	read-write	00000000h	<a href="#">Section 2.5.3.37</a>
FB0h	LAR	ITM Lock Access Register	write-only	00000000h	<a href="#">Section 2.5.3.38</a>
FB4h	LSR	ITM Lock Status Register	read-only	00000003h	<a href="#">Section 2.5.3.39</a>

### 2.5.3.1 STIM0 Register (Offset = 0h)

STIM0 is shown in [Figure 2-117](#) and described in [Table 2-130](#).

ITM Stimulus Port 0. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-117. STIM0 Register**

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

**Table 2-130. STIM0 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.2 STIM1 Register (Offset = 4h)

STIM1 is shown in [Figure 2-118](#) and described in [Table 2-131](#).

ITM Stimulus Port 1. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-118. STIM1 Register**

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

**Table 2-131. STIM1 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.3 STIM2 Register (Offset = 8h)

STIM2 is shown in [Figure 2-119](#) and described in [Table 2-132](#).

ITM Stimulus Port 2. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-119. STIM2 Register**

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

**Table 2-132. STIM2 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.4 STIM3 Register (Offset = Ch)

STIM3 is shown in [Figure 2-120](#) and described in [Table 2-133](#).

ITM Stimulus Port 3. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-120. STIM3 Register**

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

**Table 2-133. STIM3 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.5 STIM4 Register (Offset = 10h)

STIM4 is shown in [Figure 2-121](#) and described in [Table 2-134](#).

ITM Stimulus Port 4. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-121. STIM4 Register**

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

**Table 2-134. STIM4 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.6 STIM5 Register (Offset = 14h)

STIM5 is shown in [Figure 2-122](#) and described in [Table 2-135](#).

ITM Stimulus Port 5. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-122. STIM5 Register**

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

**Table 2-135. STIM5 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.7 STIM6 Register (Offset = 18h)

STIM6 is shown in [Figure 2-123](#) and described in [Table 2-136](#).

ITM Stimulus Port 6. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-123. STIM6 Register**

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

**Table 2-136. STIM6 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.8 STIM7 Register (Offset = 1Ch)

STIM7 is shown in [Figure 2-124](#) and described in [Table 2-137](#).

ITM Stimulus Port 7. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-124. STIM7 Register**

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

**Table 2-137. STIM7 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.9 STIM8 Register (Offset = 20h)

STIM8 is shown in [Figure 2-125](#) and described in [Table 2-138](#).

ITM Stimulus Port 8. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-125. STIM8 Register**

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

**Table 2-138. STIM8 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.10 STIM9 Register (Offset = 24h)

STIM9 is shown in [Figure 2-126](#) and described in [Table 2-139](#).

ITM Stimulus Port 9. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-126. STIM9 Register**

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

**Table 2-139. STIM9 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.11 STIM10 Register (Offset = 28h)

STIM10 is shown in [Figure 2-127](#) and described in [Table 2-140](#).

ITM Stimulus Port 10. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-127. STIM10 Register**

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

**Table 2-140. STIM10 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.12 STIM11 Register (Offset = 2Ch)

STIM11 is shown in [Figure 2-128](#) and described in [Table 2-141](#).

ITM Stimulus Port 11. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-128. STIM11 Register**

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

**Table 2-141. STIM11 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.13 STIM12 Register (Offset = 30h)

STIM12 is shown in [Figure 2-129](#) and described in [Table 2-142](#).

ITM Stimulus Port 12. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-129. STIM12 Register**

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

**Table 2-142. STIM12 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.14 STIM13 Register (Offset = 34h)

STIM13 is shown in [Figure 2-130](#) and described in [Table 2-143](#).

ITM Stimulus Port 13. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-130. STIM13 Register**

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

**Table 2-143. STIM13 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.15 STIM14 Register (Offset = 38h)

STIM14 is shown in [Figure 2-131](#) and described in [Table 2-144](#).

ITM Stimulus Port 14. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-131. STIM14 Register**

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

**Table 2-144. STIM14 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.16 STIM15 Register (Offset = 3Ch)

STIM15 is shown in [Figure 2-132](#) and described in [Table 2-145](#).

ITM Stimulus Port 15. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-132. STIM15 Register**

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

**Table 2-145. STIM15 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.17 STIM16 Register (Offset = 40h)

STIM16 is shown in [Figure 2-133](#) and described in [Table 2-146](#).

ITM Stimulus Port 16. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-133. STIM16 Register**

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

**Table 2-146. STIM16 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.18 STIM17 Register (Offset = 44h)

STIM17 is shown in [Figure 2-134](#) and described in [Table 2-147](#).

ITM Stimulus Port 17. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-134. STIM17 Register**

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

**Table 2-147. STIM17 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.19 STIM18 Register (Offset = 48h)

STIM18 is shown in [Figure 2-135](#) and described in [Table 2-148](#).

ITM Stimulus Port 18. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-135. STIM18 Register**

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

**Table 2-148. STIM18 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.20 STIM19 Register (Offset = 4Ch)

STIM19 is shown in [Figure 2-136](#) and described in [Table 2-149](#).

ITM Stimulus Port 19. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-136. STIM19 Register**

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

**Table 2-149. STIM19 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.21 STIM20 Register (Offset = 50h)

STIM20 is shown in [Figure 2-137](#) and described in [Table 2-150](#).

ITM Stimulus Port 20. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-137. STIM20 Register**

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

**Table 2-150. STIM20 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.22 STIM21 Register (Offset = 54h)

STIM21 is shown in [Figure 2-138](#) and described in [Table 2-151](#).

ITM Stimulus Port 21. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-138. STIM21 Register**

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

**Table 2-151. STIM21 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.23 STIM22 Register (Offset = 58h)

STIM22 is shown in [Figure 2-139](#) and described in [Table 2-152](#).

ITM Stimulus Port 22. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-139. STIM22 Register**

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

**Table 2-152. STIM22 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.24 STIM23 Register (Offset = 5Ch)

STIM23 is shown in [Figure 2-140](#) and described in [Table 2-153](#).

ITM Stimulus Port 23. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-140. STIM23 Register**

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

**Table 2-153. STIM23 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.25 STIM24 Register (Offset = 60h)

STIM24 is shown in [Figure 2-141](#) and described in [Table 2-154](#).

ITM Stimulus Port 24. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-141. STIM24 Register**

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

**Table 2-154. STIM24 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.26 STIM25 Register (Offset = 64h)

STIM25 is shown in [Figure 2-142](#) and described in [Table 2-155](#).

ITM Stimulus Port 25. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-142. STIM25 Register**

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

**Table 2-155. STIM25 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.27 STIM26 Register (Offset = 68h)

STIM26 is shown in [Figure 2-143](#) and described in [Table 2-156](#).

ITM Stimulus Port 26. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-143. STIM26 Register**

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

**Table 2-156. STIM26 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.28 STIM27 Register (Offset = 6Ch)

STIM27 is shown in [Figure 2-144](#) and described in [Table 2-157](#).

ITM Stimulus Port 27. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-144. STIM27 Register**

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

**Table 2-157. STIM27 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.29 STIM28 Register (Offset = 70h)

STIM28 is shown in [Figure 2-145](#) and described in [Table 2-158](#).

ITM Stimulus Port 28. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-145. STIM28 Register**

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

**Table 2-158. STIM28 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.30 STIM29 Register (Offset = 74h)

STIM29 is shown in [Figure 2-146](#) and described in [Table 2-159](#).

ITM Stimulus Port 29. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-146. STIM29 Register**

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

**Table 2-159. STIM29 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.31 STIM30 Register (Offset = 78h)

STIM30 is shown in [Figure 2-147](#) and described in [Table 2-160](#).

ITM Stimulus Port 30. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set. Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-147. STIM30 Register**

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

**Table 2-160. STIM30 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.32 STIM31 Register (Offset = 7Ch)

STIM31 is shown in [Figure 2-148](#) and described in [Table 2-161](#).

ITM Stimulus Port 31. Each of the 32 stimulus ports has its own address. A write to one of these locations causes data to be written into the FIFO if the corresponding bit in the Trace Enable Register is set.

Reading from any of the stimulus ports returns the FIFO status in bit [0]: 0 = full, 1 = not full. The polled FIFO interface does not provide an atomic read-modify-write, so you must use the Cortex-M4 exclusive monitor if a polled printf is used concurrently with ITM usage by interrupts or other threads.

**Figure 2-148. STIM31 Register**

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

**Table 2-161. STIM31 Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMULUS	W	Undefined	Data write to the Stimulus Port FIFO for forwarding as a software event packet. The write is ignored if the Stimulus Port is disabled by the Trace Enable Register.
0	FIFOREADY	R	Undefined	1 = Stimulus Port FIFO can accept at least one word 0 = Stimulus Port FIFO full or when Stimulus Port is disabled by the Trace Enable Register

### 2.5.3.33 TER Register (Offset = E00h) [reset = 00000000h]

TER is shown in [Figure 2-149](#) and described in [Table 2-162](#).

ITM Trace Enable Register. Use the Trace Enable Register to generate trace data by writing to the corresponding stimulus port. Note: Privileged writes are accepted to this register if ITMENA is set. User writes are accepted to this register if ITMENA is set and the appropriate privilege mask is cleared. Privileged access to the stimulus ports enables an RTOS kernel to ensure instrumentation slots or bandwidth as required.

**Figure 2-149. TER Register**

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
STIMENA																															
R/W-0h																															

**Table 2-162. TER Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	STIMENA	R/W	0h	Bit mask to enable tracing on ITM stimulus ports. One bit per stimulus port.

### 2.5.3.34 TPR Register (Offset = E40h) [reset = 00000000h]

TPR is shown in [Figure 2-150](#) and described in [Table 2-163](#).

ITM Trace Privilege Register. Use the ITM Trace Privilege Register to enable an operating system to control which stimulus ports are accessible by user code. Note: You can only write to this register in privileged mode.

**Figure 2-150. TPR Register**

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

**Table 2-163. TPR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-4	RESERVED	R/W	0h	
3-0	PRIVMASK	R/W	0h	Bit mask to enable tracing on ITM stimulus ports: bit [0] = stimulus ports [7:0], bit [1] = stimulus ports [15:8], bit [2] = stimulus ports [23:16], bit [3] = stimulus ports [31:24].

### 2.5.3.35 TCR Register (Offset = E80h) [reset = 00000000h]

TCR is shown in [Figure 2-151](#) and described in [Table 2-164](#).

ITM Trace Control Register. Use this register to configure and control ITM transfers. Note 1: You can only write to this register in privilege mode. Note 2: DWT is not enabled in the ITM block. However, DWT stimulus entry into the FIFO is controlled by DWTENA. If DWT requires timestamping, the TSSEN bit must be set.

**Figure 2-151. TCR Register**

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
BUSY			ATBID				
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED						TSPRESCALE	
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED			SWOENA	DWTENA	SYNCENA	TSENA	ITMENA
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

**Table 2-164. TCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-24	RESERVED	R/W	0h	
23	BUSY	R/W	0h	Set when ITM events present and being drained.
22-16	ATBID	R/W	0h	ATB ID for CoreSight system.
15-10	RESERVED	R/W	0h	
9-8	TSPRESCALE	R/W	0h	TSPrescale Timestamp prescaler. 0b (R/W) = no prescaling 1b (R/W) = divide by 4 10b (R/W) = divide by 16 11b (R/W) = divide by 64
7-5	RESERVED	R/W	0h	
4	SWOENA	R/W	0h	Enables asynchronous clocking of the timestamp counter.
3	DWTENA	R/W	0h	Enables the DWT stimulus.
2	SYNCENA	R/W	0h	Enables sync packets for TPIU.
1	TSENA	R/W	0h	Enables differential timestamps. Differential timestamps are emitted when a packet is written to the FIFO with a non-zero timestamp counter, and when the timestamp counter overflows. Timestamps are emitted during idle times after a fixed number of two million cycles. This provides a time reference for packets and inter-packet gaps. If SWOENA (bit [4]) is set, timestamps are triggered by activity on the internal trace bus only. In this case there is no regular timestamp output when the ITM is idle.
0	ITMENA	R/W	0h	Enable ITM. This is the master enable, and must be set before ITM Stimulus and Trace Enable registers can be written.

### 2.5.3.36 IWR Register (Offset = EF8h) [reset = 00000000h]

IWR is shown in [Figure 2-152](#) and described in [Table 2-165](#).

ITM Integration Write Register. Use this register to determine the behavior of the ATVALIDM bit. Note: Bit [0] drives ATVALIDM when mode is set.

**Figure 2-152. IWR Register**

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

**Table 2-165. IWR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	W	0h	
0	ATVALIDM	W	0h	When the integration mode is set: 0 = ATVALIDM clear. 1 = ATVALIDM set. 0b (R/W) = ATVALIDM clear 1b (R/W) = ATVALIDM set

### 2.5.3.37 IMCR Register (Offset = F00h) [reset = 00000000h]

IMCR is shown in [Figure 2-153](#) and described in [Table 2-166](#).

ITM Integration Mode Control Register. Use this register to enable write accesses to the Control Register.

**Figure 2-153. IMCR Register**

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

**Table 2-166. IMCR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R/W	0h	
0	INTEGRATION	R/W	0h	

### 2.5.3.38 LAR Register (Offset = FB0h) [reset = 00000000h]

LAR is shown in [Figure 2-154](#) and described in [Table 2-167](#).

ITM Lock Access Register. Use this register to prevent write accesses to the Control Register.

**Figure 2-154. LAR Register**

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
LOCK_ACCESS																															
W-0h																															

**Table 2-167. LAR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	LOCK_ACCESS	W	0h	A privileged write of 0xC5ACCE55 enables more write access to Control Register 0xE00::0xFFC. An invalid write removes write access.

### 2.5.3.39 LSR Register (Offset = FB4h) [reset = 00000003h]

LSR is shown in [Figure 2-155](#) and described in [Table 2-168](#).

ITM Lock Status Register. Use this register to enable write accesses to the Control Register.

**Figure 2-155. LSR Register**

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				BYTEACC	ACCESS	PRESENT	
R-0h				R-0h	R-1h	R-1h	

**Table 2-168. LSR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	
2	BYTEACC	R	0h	You cannot implement 8-bit lock accesses.
1	ACCESS	R	1h	Write access to component is blocked. All writes are ignored, reads are permitted.
0	PRESENT	R	1h	Indicates that a lock mechanism exists for this component.

## **Reset Controller (*RSTCTL*)**

This chapter describes the Reset Controller in MSP432P4xx devices.

Topic	Page
3.1 <b>Introduction</b> .....	<b>245</b>
3.2 <b>Reset Classification</b> .....	<b>245</b>
3.3 <b>RSTCTL Registers</b> .....	<b>248</b>

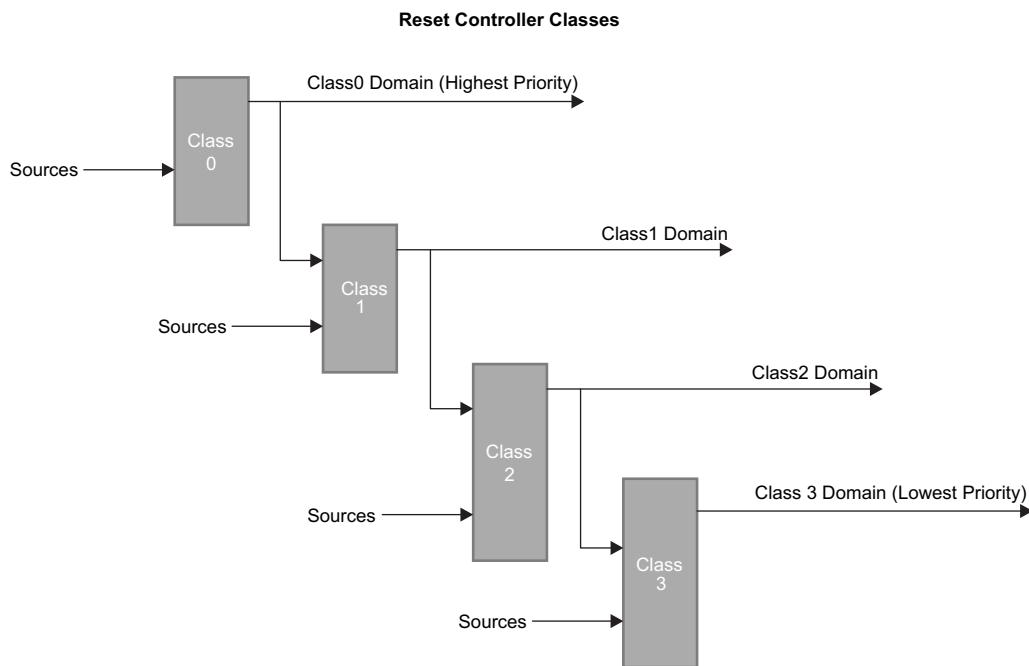
### 3.1 Introduction

The Reset Controller is a module that collates inputs from all of the various sources of reset in the device and generates different classes of resets that are fed back to the device.

### 3.2 Reset Classification

The MSP432P4xx device contains resets of different classes, with each class resulting in a different initialization state of the application registers. The resets are classified based on the user application view of the device, and the extent of control required by the same over the state of the device. In different use case conditions, either during application development, code debug, or real-time application execution, a different class of reset may be desired to gain control over the device, without completely sacrificing the device state.

**Figure 3-1** shows the reset generation mechanism with its classes of resets. The reset priority is in decreasing order from left to right, meaning that each reset automatically initiates all lower priority resets (if any). However, a reset class of lower priority does not trigger a reset of higher priority. The nomenclature of the reset classes is described in the following subsections.



**Figure 3-1. Reset Classes**

#### 3.2.1 Class 0 : Power On/Off Reset (POR) Class

A 'Power On/Off Reset' (POR) refers to any type of reset that can help gain control of a device in a completely uninitialized (or random) state. A POR may be required by the device in any of the following cases:

- A true power-on or power-off condition (application or removal of power to the device)
- A 'voltage exception' condition that is generated by the power supply system (PSS). This condition can be caused by the supervision logic for either the  $V_{CC}$  or core domain voltage.
- Exit from LPM3.5 or LPM4.5 modes of operation (initiated by the PCM)
- A user-driven full chip reset. This reset can be initiated either through the RSTn pin, through the debugger, through the SYSCTL.
- DCO short-circuit fault in external resistor mode of operation.

From the user application perspective, all of the above sources result in the same reset state, and are hence classified under one reset category, which is termed as POR. The status of the device on a POR is as follows

- All components in the device are reset.
- Debugger loses connection to the device and control of it.
- The device undergoes a full reboot.
- On-chip SRAM values are not ensured to be retained.

---

**NOTE:** Although all the above mentioned sources result in a POR class reset, the reset recovery time from each may be different (due to settling time requirements of analog components). Refer to the device specific datasheet for details of the recovery latency from the different POR sources.

---

**NOTE:** Refer to device specific datasheet for details on the available POR sources.

---

### 3.2.2 Class 1 : Reboot Reset

The Reboot Reset is identical to the POR from a device application perspective, except that it does not reset the debug components of the CPU. This is a special class of reset that is initiated only through software control. The Reboot Reset is a way of emulating the complete startup of the device (typically through a POR) without changing or resetting the boot mode of the device. This class of reset allows the user to force a 'controlled' re-execution of the boot code without the need to issue a complete POR. As a result, the debugger or user application can request a boot-override mode of operation (like requesting for securing a section of code). These boot override mechanisms are specified in the SYSCTL chapter.

---

**NOTE:** A Reboot Reset is activated whenever a reset that is higher in class is active.

---

### 3.2.3 Class 2 : Hard Reset

A Hard Reset is initiated under user application control and is a deterministic event. This means that the device is already in a known state, and developer or application wishes to re-initialize the system as a reaction to a particular event or condition. Application driven requirement to restart without rebooting. This may be due to a catastrophic event detected by application, or a debug scenario (developer wishing to restart the application through the debugger or through the BSL)

From an application perspective, a Hard Reset does the following:

- Resets the processor and all application configured peripherals in the system. This includes the bus system, thereby aborting any pending bus transactions.
- Returns control to the user code.
- Debugger connection to the device is maintained.
- It does NOT reboot the device.
- On-chip SRAM values are retained.

---

**NOTE:** The Hard Reset class sets status flag registers that report the exact source of the Hard Reset. The application can use these registers to select the necessary course of action. Refer to device specific datasheet for details on the available Hard Reset sources.

---

**NOTE:** A Hard Reset is activated whenever a reset that is higher in class is active.

---

### 3.2.4 Class 3 : Soft Reset

A Soft Reset is initiated under user application control and is a deterministic event. This class resets only the execution-related components of the system. All other application related configuration is maintained, thereby retaining the application's view of the device. Peripherals that are configured by the application continue their operation through a Soft Reset.

From an application perspective, a Soft Reset has the following implications:

- Resets the following execution related components in the system:
  - SYSRESETn of the Cortex-M4. All bus transactions in the M4 (except the debug PPB space) are aborted.
  - WDT module.
- All system-level bus transactions are maintained.
- All peripheral configurations are maintained.
- Returns control to the user code.
- Debugger connection to the device is maintained.
- It does NOT reboot the device.
- On-chip SRAM values are retained

---

**NOTE:** The Soft Reset class sets status flag registers that report the exact source of the Soft Reset. An application can use these registers to select the necessary course of action. Refer to device specific datasheet for details on the available Soft Reset sources.

---

---

**NOTE:** A Soft Reset is activated whenever a reset that is higher in class is active.

---

---

**NOTE:** The Cortex-M4 processor can initiate an 'processor only' Reset using the VECTRESET bit in the Cortex-M4 Application Interrupt and Reset Control (AIRCR) Register. This reset is not visible outside of the M4, and the user application must manage this condition when using this reset.

Alternatively, the SYSRESETREQ bit of the same register can be used to generate a Hard Reset on the device.

---

### 3.3 RSTCTL Registers

**Table 3-1. RSTCTL Registers**

Offset	Acronym	Register Name	Section
000h	RSTCTL_RESET_REQ	Reset Request Register	<a href="#">Section 3.3.1</a>
004h	RSTCTL_HARDRESET_STAT	Hard Reset Status Register	<a href="#">Section 3.3.2</a>
008h	RSTCTL_HARDRESET_CLR	Hard Reset Status Clear Register	<a href="#">Section 3.3.3</a>
00Ch	RSTCTL_HARDRESET_SET	Hard Reset Status Set Register	<a href="#">Section 3.3.4</a>
010h	RSTCTL_SOFTRESET_STAT	Soft Reset Status Register	<a href="#">Section 3.3.5</a>
014h	RSTCTL_SOFTRESET_CLR	Soft Reset Status Clear Register	<a href="#">Section 3.3.6</a>
018h	RSTCTL_SOFTRESET_SET	Soft Reset Status Set Register	<a href="#">Section 3.3.7</a>
100h	RSTCTL_PSSRESET_STAT	PSS Reset Status Register	<a href="#">Section 3.3.8</a>
104h	RSTCTL_PSSRESET_CLR	PSS Reset Status Clear Register	<a href="#">Section 3.3.9</a>
108h	RSTCTL_PCMRESET_STAT	PCM Reset Status Register	<a href="#">Section 3.3.10</a>
10Ch	RSTCTL_PCMRESET_CLR	PCM Reset Status Clear Register	<a href="#">Section 3.3.11</a>
110h	RSTCTL_PINRESET_STAT	Pin Reset Status Register	<a href="#">Section 3.3.12</a>
114h	RSTCTL_PINRESET_CLR	Pin Reset Status Clear Register	<a href="#">Section 3.3.13</a>
118h	RSTCTL_REBOOTRESET_STAT	Reboot Reset Status Register	<a href="#">Section 3.3.14</a>
11Ch	RSTCTL_REBOOTRESET_CLR	Reboot Reset Status Clear Register	<a href="#">Section 3.3.15</a>
120h	RSTCTL_CSRESET_STAT	CS Reset Status Register	<a href="#">Section 3.3.16</a>
124h	RSTCTL_CSRESET_CLR	CS Reset Status Clear Register	<a href="#">Section 3.3.17</a>

---

**NOTE:** This is a 32-bit module and can be accessed ONLY through word (32-bit) access.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### **3.3.1 RSTCTL\_RESET\_REQ Register (offset = 00h)**

Reset Request Register

This register controls software driven reset requests for the Hard and Soft Classes.

**Figure 3-2. RSTCTL\_RESET\_REQ Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSTKEY															
w	w	w	w	w	w	w	w	r	r	r	r	r	r	w	w

**Table 3-2. RSTCTL\_RESET\_REQ Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads 0h
15-8	RSTKEY	W	0h	Must be written with 69h to enable writes to bits 1-0 (in the same write operation), else writes to bits 1-0 are ignored
7-2	Reserved	R	0h	Reserved. Always reads 0h
1	HARD_REQ	W	0h	If written with 1, generates a Hard Reset request to the Reset Controller
0	SOFT_REQ	W	0h	If written with 1, generates a Soft Reset request to the Reset Controller

### 3.3.2 RSTCTL\_HARDRESET\_STAT Register (offset = 04h)

Hard Reset Status Register

**Figure 3-3. RSTCTL\_HARDRESET\_STAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC15	SRC14	SRC13	SRC12	SRC11	SRC10	SRC9	SRC8	SRC7	SRC6	SRC5	SRC4	SRC3	SRC2	SRC1	SRC0
r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)

**Table 3-3. RSTCTL\_HARDRESET\_STAT Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads 0h
15	SRC15	R	0h	If 1, indicates that SRC15 was the source of the Hard Reset <sup>(1)</sup>
14	SRC14	R	0h	If 1, indicates that SRC14 was the source of the Hard Reset <sup>(1)</sup>
13	SRC13	R	0h	If 1, indicates that SRC13 was the source of the Hard Reset <sup>(1)</sup>
12	SRC12	R	0h	If 1, indicates that SRC12 was the source of the Hard Reset <sup>(1)</sup>
11	SRC11	R	0h	If 1, indicates that SRC11 was the source of the Hard Reset <sup>(1)</sup>
10	SRC10	R	0h	If 1, indicates that SRC10 was the source of the Hard Reset <sup>(1)</sup>
9	SRC9	R	0h	If 1, indicates that SRC9 was the source of the Hard Reset <sup>(1)</sup>
8	SRC8	R	0h	If 1, indicates that SRC8 was the source of the Hard Reset <sup>(1)</sup>
7	SRC7	R	0h	If 1, indicates that SRC7 was the source of the Hard Reset <sup>(1)</sup>
6	SRC6	R	0h	If 1, indicates that SRC6 was the source of the Hard Reset <sup>(1)</sup>
5	SRC5	R	0h	If 1, indicates that SRC5 was the source of the Hard Reset <sup>(1)</sup>
4	SRC4	R	0h	If 1, indicates that SRC4 was the source of the Hard Reset <sup>(1)</sup>
3	SRC3	R	0h	If 1, indicates that SRC3 was the source of the Hard Reset <sup>(1)</sup>
2	SRC2	R	0h	If 1, indicates that SRC2 was the source of the Hard Reset <sup>(1)</sup>
1	SRC1	R	0h	If 1, indicates that SRC1 was the source of the Hard Reset <sup>(1)</sup>
0	SRC0	R	0h	If 1, indicates that SRC0 was the source of the Hard Reset <sup>(1)</sup>

<sup>(1)</sup> Refer to the device-specific data sheet for the mapping of device-level Hard Reset sources to the appropriate bit in this register.

### 3.3.3 RSTCTL\_HARDRESET\_CLR Register (offset = 08h)

Hard Reset Status Clear Register

**Figure 3-4. RSTCTL\_HARDRESET\_CLR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC15	SRC14	SRC13	SRC12	SRC11	SRC10	SRC9	SRC8	SRC7	SRC6	SRC5	SRC4	SRC3	SRC2	SRC1	SRC0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

**Table 3-4. RSTCTL\_HARDRESET\_CLR Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads 0h
15	SRC15	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
14	SRC14	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
13	SRC13	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
12	SRC12	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
11	SRC11	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
10	SRC10	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
9	SRC9	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
8	SRC8	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
7	SRC7	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
6	SRC6	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
5	SRC5	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
4	SRC4	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
3	SRC3	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
2	SRC2	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
1	SRC1	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.
0	SRC0	W	0h	Write 1 clears the corresponding bit in the RSTCTL_HARDRESET_STAT. Write 0 has no effect.

### 3.3.4 RSTCTL\_HARDRESET\_SET Register (offset = 0Ch)

Hard Reset Status Set Register

**Figure 3-5. RSTCTL\_HARDRESET\_SET Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC15	SRC14	SRC13	SRC12	SRC11	SRC10	SRC9	SRC8	SRC7	SRC6	SRC5	SRC4	SRC3	SRC2	SRC1	SRC0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

**Table 3-5. RSTCTL\_HARDRESET\_SET Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads 0h
15	SRC15	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
14	SRC14	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
13	SRC13	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
12	SRC12	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
11	SRC11	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
10	SRC10	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
9	SRC9	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
8	SRC8	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
7	SRC7	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
6	SRC6	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
5	SRC5	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
4	SRC4	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
3	SRC3	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
2	SRC2	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
1	SRC1	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.
0	SRC0	W	0h	Write 1 sets the corresponding bit in the RSTCTL_HARDRESET_STAT (and initiates a Hard Reset). Write 0 has no effect.

### 3.3.5 RSTCTL\_SOFTRESET\_STAT Register (offset = 10h)

Soft Reset Status Register

**Figure 3-6. RSTCTL\_SOFTRESET\_STAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC15	SRC14	SRC13	SRC12	SRC11	SRC10	SRC9	SRC8	SRC7	SRC6	SRC5	SRC4	SRC3	SRC2	SRC1	SRC0
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

**Table 3-6. RSTCTL\_SOFTRESET\_STAT Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads 0h
15	SRC15	R	0h	If 1, indicates that SRC15 was the source of the Soft Reset <sup>(1)</sup>
14	SRC14	R	0h	If 1, indicates that SRC14 was the source of the Soft Reset <sup>(1)</sup>
13	SRC13	R	0h	If 1, indicates that SRC13 was the source of the Soft Reset <sup>(1)</sup>
12	SRC12	R	0h	If 1, indicates that SRC12 was the source of the Soft Reset <sup>(1)</sup>
11	SRC11	R	0h	If 1, indicates that SRC11 was the source of the Soft Reset <sup>(1)</sup>
10	SRC10	R	0h	If 1, indicates that SRC10 was the source of the Soft Reset <sup>(1)</sup>
9	SRC9	R	0h	If 1, indicates that SRC9 was the source of the Soft Reset <sup>(1)</sup>
8	SRC8	R	0h	If 1, indicates that SRC8 was the source of the Soft Reset <sup>(1)</sup>
7	SRC7	R	0h	If 1, indicates that SRC7 was the source of the Soft Reset <sup>(1)</sup>
6	SRC6	R	0h	If 1, indicates that SRC6 was the source of the Soft Reset <sup>(1)</sup>
5	SRC5	R	0h	If 1, indicates that SRC5 was the source of the Soft Reset <sup>(1)</sup>
4	SRC4	R	0h	If 1, indicates that SRC4 was the source of the Soft Reset <sup>(1)</sup>
3	SRC3	R	0h	If 1, indicates that SRC3 was the source of the Soft Reset <sup>(1)</sup>
2	SRC2	R	0h	If 1, indicates that SRC2 was the source of the Soft Reset <sup>(1)</sup>
1	SRC1	R	0h	If 1, indicates that SRC1 was the source of the Soft Reset <sup>(1)</sup>
0	SRC0	R	0h	If 1, indicates that SRC0 was the source of the Soft Reset <sup>(1)</sup>

<sup>(1)</sup> Refer to the device-specific data sheet for the mapping of device-level Soft Reset sources to the appropriate bit in this register.

### 3.3.6 RSTCTL\_SOFTRESET\_CLR Register (offset = 14h)

Soft Reset Status Clear Register

**Figure 3-7. RSTCTL\_SOFTRESET\_CLR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC15	SRC14	SRC13	SRC12	SRC11	SRC10	SRC9	SRC8	SRC7	SRC6	SRC5	SRC4	SRC3	SRC2	SRC1	SRC0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

**Table 3-7. RSTCTL\_SOFTRESET\_CLR Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads 0h
15	SRC15	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
14	SRC14	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
13	SRC13	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
12	SRC12	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
11	SRC11	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
10	SRC10	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
9	SRC9	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
8	SRC8	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
7	SRC7	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
6	SRC6	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
5	SRC5	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
4	SRC4	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
3	SRC3	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
2	SRC2	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
1	SRC1	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.
0	SRC0	W	0h	Write 1 clears the corresponding bit in the RSTCTL_SOFTRESET_STAT. Write 0 has no effect.

### 3.3.7 RSTCTL\_SOFTRESET\_SET Register (offset = 18h)

Soft Reset Status Set Register

**Figure 3-8. RSTCTL\_SOFTRESET\_SET Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC15	SRC14	SRC13	SRC12	SRC11	SRC10	SRC9	SRC8	SRC7	SRC6	SRC5	SRC4	SRC3	SRC2	SRC1	SRC0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

**Table 3-8. RSTCTL\_SOFTRESET\_SET Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads 0h
15	SRC15	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
14	SRC14	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
13	SRC13	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
12	SRC12	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
11	SRC11	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
10	SRC10	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
9	SRC9	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
8	SRC8	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
7	SRC7	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
6	SRC6	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
5	SRC5	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
4	SRC4	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
3	SRC3	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
2	SRC2	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
1	SRC1	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.
0	SRC0	W	0h	Write 1 sets the corresponding bit in the RSTCTL_SOFTRESET_STAT (and initiates a Soft Reset). Write 0 has no effect.

### 3.3.8 RSTCTL\_PSSRESET\_STAT Register (offset = 100h)

PSS Reset Status Register

**Figure 3-9. RSTCTL\_PSSRESET\_STAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
														r-[1]	r-[1]
														r-[1]	r-[1]

**Table 3-9. RSTCTL\_PSSRESET\_STAT Register Description**

Bit	Field	Type	Reset	Description
31-4	Reserved	R	0h	Reserved. Always reads 0h
3	VCCDET	R	1h	Indicates if POR was caused by a VCCDET trip condition in the PSS
2	BGREF	R	1h	Indicates if POR was caused by a Band Gap Reference not okay condition in the PSS
1	SVSMH	R	1h	Indicates if POR was caused by an SVSMH trip condition in the PSS (with SVSMH enabled and set as supervisor)
0	Reserved	R	1h	Reserved.

**NOTE:** The bits in this register may be set in different combinations, indicating the sequence of reset events caused by the PSS. Software must evaluate the MSB of these bits to determine the nature of the PSS reset. For example, the value 1xxxb typically indicates a device power-up condition, which automatically causes all of the conditions to activate and then deassert in succession. As another example, the value 0010b indicates that the PSS generated a reset only due to SVSMH voltage threshold violation.

### 3.3.9 RSTCTL\_PSSRESET\_CLR Register (offset = 104h)

PSS Reset Status Clear Register

**Figure 3-10. RSTCTL\_PSSRESET\_CLR Register**

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

**Table 3-10. RSTCTL\_PSSRESET\_CLR Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved. Always reads 0h
0	CLR	W	0h	Write 1 clears all PSS Reset Flags in the RSTCTL_PSSRESET_STAT

### **3.3.10 RSTCTL\_PCMRESET\_STAT Register (offset = 108h)**

PCM Reset Status Register

**Figure 3-11. RSTCTL\_PCMRESET\_STAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r(0)	r(0)

**Table 3-11. RSTCTL\_PCMRESET\_STAT Register Description**

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0h	Reserved. Always reads 0h.
1	LPM45	R	0h	Indicates if POR was caused by PCM due to an exit from LPM4.5
0	LPM35	R	0h	Indicates if POR was caused by PCM due to an exit from LPM3.5

**NOTE:** The bits in this register are cleared on a POR event of RSTn/NMI or Debug POR. They are set only on the LPM3.5 and LPM4.5 exit cases and should be cleared using the CLR bit in RSTCTL\_PCMRESET\_CLR.

### **3.3.11 RSTCTL\_PCMRESET\_CLR Register (offset = 10Ch)**

PCM Reset Status Clear Register

**Figure 3-12. RSTCTL\_PCMRESET\_CLR Register**

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

**Table 3-12. RSTCTL\_PCMRESET\_CLR Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved. Always reads 0h
0	CLR	W	0h	Write 1 clears all PCM Reset Flags in the RSTCTL_PCMRESET_STAT

### 3.3.12 RSTCTL\_PINRESET\_STAT Register (offset = 110h)

Pin Reset Status Register

**Figure 3-13. RSTCTL\_PINRESET\_STAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r-10/

**Table 3-13. RSTCTL\_PINRESET\_STAT Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved. Always reads 0h
0	RSTNMI	R	0h	Indicates if POR was caused by RSTn/NMI pin based reset event in the device. This bit comes up by default as 0 in the case of cold power up. This is only set if there is a POR caused due to the RSTn/NMI pin.

### 3.3.13 RSTCTL\_PINRESET\_CLR Register (offset = 114h)

Pin Reset Status Clear Register

**Figure 3-14. RSTCTL\_PINRESET\_CLR Register**

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

**Table 3-14. RSTCTL\_PINRESET\_CLR Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved. Always reads 0h
0	CLR	W	0h	Write 1 clears the RSTn/NMI Pin Reset Flag in RSTCTL_PINRESET_STAT

### **3.3.14 RSTCTL\_REBOOTRESET\_STAT Register (offset = 118h)**

Reboot Reset Status Register

**Figure 3-15. RSTCTL\_REBOOTRESET\_STAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r-(0)

**Table 3-15. RSTCTL\_REBOOTRESET\_STAT Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved. Always reads 0h
0	REBOOT	R	0h	Indicates if Reboot reset was caused by the SYSCTL module.

### **3.3.15 RSTCTL\_REBOOTRESET\_CLR Register (offset = 11Ch)**

Reboot Reset Status Clear Register

**Figure 3-16. RSTCTL\_REBOOTRESET\_CLR Register**

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

**Table 3-16. RSTCTL\_REBOOTRESET\_CLR Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved. Always reads 0h
0	CLR	W	0h	Write 1 clears the Reboot Reset Flag in RSTCTL_REBOOTRESET_STAT

### 3.3.16 RSTCTL\_CSRESET\_STAT Register (offset = 120h)

CS Reset Status Register

**Figure 3-17. RSTCTL\_CSRESET\_STAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r(0)

**Table 3-17. RSTCTL\_CSRESET\_STAT Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved. Always reads 0h.
0	DCOR_SHT	R	0h	Indicates if POR was caused by DCO short circuit fault in the external resistor mode

### 3.3.17 RSTCTL\_CSRESET\_CLR Register (offset = 124h)

CS Reset Status Clear Register

**Figure 3-18. RSTCTL\_CSRESET\_CLR Register**

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

**Table 3-18. RSTCTL\_CSRESET\_CLR Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved. Always reads 0h.
0	CLR	W	0h	Write 1 clears the DCOR_SHT Flag in RSTCTL_CSRESET_STAT as well as DCOR_SHTIFG flag in CSIFG register of clock system

## System Controller (SYSCTL)

This chapter describes the System Controller on MSP432P4xx devices.

Topic	Page
4.1 SYSCTL Introduction .....	262
4.2 Device Memory Configuration and Status.....	262
4.3 NMI Configuration.....	263
4.4 Watchdog Timer Reset Configuration .....	263
4.5 Peripheral Halt Control.....	263
4.6 Glitch Filtering on Digital I/Os .....	263
4.7 Reset Status and Override Control .....	264
4.8 Device Security .....	264
4.9 Device Descriptor Table .....	279
4.10 ARM Cortex-M4F ROM Table Based Part Number .....	284
4.11 SYSCTL Registers .....	285

## 4.1 SYSCTL Introduction

The System Controller (SYSCTL) on MSP432P4xx devices is responsible for the following functions:

- Device memory configuration and status
- NMI sources configuration and status
- Watchdog configuration to generate hard reset or soft reset
- Clock run or stop configuration to various modules in debug mode
- Override controls for resets for device debug
- Device security configuration
- Device configuration and peripherals calibration information through Device Descriptors

## 4.2 Device Memory Configuration and Status

The following sections provide information on device memory configuration and status provided by SYSCTL.

### 4.2.1 Flash

The size of flash main memory can vary from device to device in the MSP432P4xx family. The flash main memory can be viewed as two independent identical banks, each of which is half of the total size of the flash main memory. This allows simultaneous read or execute from one bank while the other bank is undergoing a program or erase operation. The size of flash main memory available on the device is represented through the SYS\_FLASH\_SIZE register.

### 4.2.2 SRAM

The size of SRAM can vary from device to device in the MSP432P4xx family. The SRAM is aliased in both Code and SRAM memory zones of the CPU memory map. The size of SRAM available on the device is reported in the SYS\_SRAM\_SIZE register.

#### 4.2.2.1 SRAM Bank Enable Configuration

The application can optimize the power consumption of the SRAM. To enable this optimization, the SRAM memory is divided into different banks that can be powered down individually. Banks that are powered down remain powered down in both active and low-power modes of operation, thereby limiting any unnecessary inrush current when the device transitions between active and retention-based low-power modes. The application can also disable one (or more) banks for a certain stage in the processing and re-enable it for another stage.

When a particular bank is disabled, reads to its address space return 0h, and writes are discarded. To prevent 'holes' in the memory map, if a particular bank is enabled, all of the lower banks are also forced to enabled state. This ensures a contiguous memory map through the set of enabled banks instead of allowing a disabled bank to appear between enabled banks. For example:

- If there are eight banks in the device, values of 00111111 and 00000111 are acceptable.
- Values like 00010111 are not valid, and the resultant bank configuration is automatically set to 00011111.
- For example, for a 8-bank SRAM, the only allowed values are 00000001, 00000011, 00000111, 00001111, 00011111, 00111111, 01111111 and 11111111.

**Bank0 of SRAM is always enabled and cannot be disabled.** For all other banks, any enable or disable change results in the SRAM\_RDY bit of the SYS\_SRAM\_BANKEN register being set to 0 until the configuration change is effective. Any accesses to the SRAM is stalled during this time, and access resumes only after the SRAM banks are ready for read or write operations. This is handled transparently and does not require any code intervention

#### 4.2.2.2 SRAM Bank Retention Configuration and Backup Memory

The application can optimize the leakage power consumption of the SRAM in LPM3 and LPM4 modes of operation. To enable this, each SRAM bank can be individually configured for retention. Banks that are enabled for retention retain their data through the LPM3 and LPM4 modes. The application can also retain a subset of the enabled banks. The Bank Retention Configuration for any bank have an effect only when that bank is enabled using the SRAM Bank Enable Configuration is set.

For example, the application may need 32KB of SRAM for its processing needs (4 banks are kept enabled). However, of these four banks, only one bank may contain critical data that must be retained in LPM3 or LPM4, while the rest are powered off completely to minimize power consumption. See SYS\_SRAM\_BANKRET register for details on how individual banks can be controlled by the application.

**Bank0 of SRAM is always retained and cannot be powered down.** Therefore, it also operates up as a possible backup memory in the LPM3, LPM4, and LPM3.5 modes of operation.

### 4.3 NMI Configuration

The following NMI sources are available on MSP432P4xx devices.

- RSTn/NMI device pin in NMI configuration
- Clock System (CS) sources
- Power Supply System (PSS) sources
- Power Control Manager (PCM) sources

See SYS\_NMI\_CTLSTAT register for configuration and status of different NMI sources available on the device. These NMI sources can also be configured as maskable interrupts through appropriate programming of the NVIC registers.

### 4.4 Watchdog Timer Reset Configuration

The watchdog timer module generates a reset upon password violation or timeout while in watchdog timer mode of operation. It is possible to configure these watchdog timer module reset sources to hard reset or soft reset independently through SYS\_WDTRESET\_CTL register.

### 4.5 Peripheral Halt Control

The Peripheral Halt Control (SYS\_PERIHALT\_CTL) register in the System Controller module allows the user independent control over the functionality of device peripherals during code development and debug. When the CPU is halted, the bits in this register can control whether the corresponding peripheral freezes its operation (such as incrementing, transmit, and receive) or continues its operation (debug remains nonintrusive). The registers of the peripheral remain accessible irrespective of the value programmed in SYS\_PERIHALT\_CTL Register.

### 4.6 Glitch Filtering on Digital I/Os

Some of the interrupt and wake-up capable digital I/Os can suppress glitches through the use of analog glitch filter to prevent unintentional interrupt or wake-up during device operation. The analog filter suppresses a minimum of 250-ns wide glitches. The glitch filter on these selected digital I/Os is enabled by default. If the glitch filtering capability is not required in the application, it can be bypassed using the SYS\_DIO\_GLTFILT\_CTL register. When GLTFLT\_EN bit in this register is cleared, the glitch filters on all the digital I/Os are bypassed. The glitch filter is automatically bypassed on a digital I/O when it is configured for peripheral or analog functionality by programming the respective PySEL0.x and PySEL1.x registers.

## 4.7 Reset Status and Override Control

The SYSCTL module has registers to monitor the status of the various classes of resets in the device. In addition, it can override device resets and initiate reset requests for debug purposes. See [Section 4.11](#) for details.

---

**NOTE:** Reset overrides are for debug of application code and take effect only when the device security (JTAG and SWD lock or IP protection) is inactive.

---

## 4.8 Device Security

This section talks about the device security control options and how to setup device security within the MSP432P4xx device family.

### 4.8.1 Device Security Introduction

One of the most important functions of the SYSCTL module is the security control of the device. The SYSCTL enables the capability to secure the device against accesses from the debugger (JTAG and SWD lock feature). In addition, the SYSCTL enables security control for different configurable zones of the device (IP protection feature). The application can load a secure piece of code (IP software/middleware) into the device flash memory and configure that zone of memory as secure. This section deals with how the application can set up the device for the various device security options.

### 4.8.2 Device Security Components

The SYSCTL module achieves device security by interacting with the following components on the device.

- Device boot-code
- Flash/JTAG mailbox mechanism for application level interaction with device boot-code

### 4.8.3 JTAG and SWD Lock Based Security

The SYSCTL module provides provisions to block the debugger accesses to the device through the JTAG or SWD interfaces. This feature is called JTAG and SWD lock. To setup JTAG and SWD lock, application is required to initiate a boot override sequence in the system. See [Section 4.8.6](#) for details on boot overrides. Post a successful boot override sequence, only a few registers of SYSCTL are accessible through the JTAG and SWD interface. All other debug accesses are blocked through these interfaces.

Access into a JTAG and SWD locked device can be re-enabled through initiation of a factory reset boot override request through the SYSCTL registers and Flash boot-override mailbox.

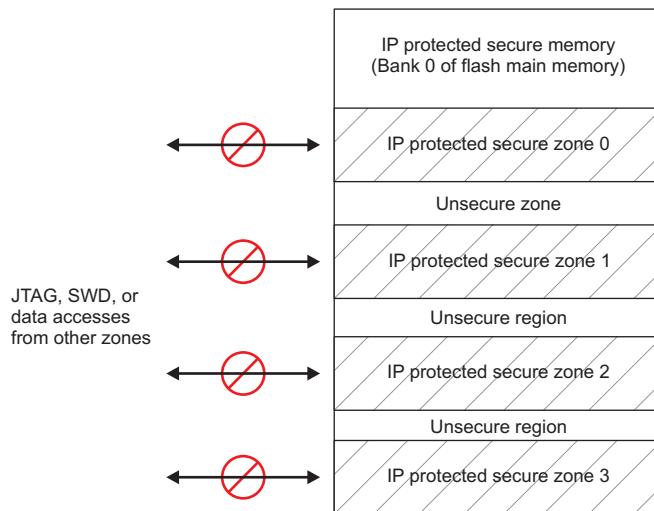
### 4.8.4 IP Protection Through Secure Memory Zones

If IP protection is deployed, sections of the Bank 0 of flash main memory can be configured as 'secure zones', allowing single or multiple vendors to store sensitive or proprietary data in these zones. IP protection permits code development and debug without compromising certain other regions of the flash which may already be containing secured code from a different source. Any access to an IP protected secure memory zone is filtered based on the following criteria:

- Instruction Fetches to an IP protected secure zone are always permitted
- Data fetches from the secure zone are permitted only if both of the following conditions are satisfied
  - Instruction causing the data fetch lies within the same secure zone
  - The secure zone being accessed has been unlocked for data accesses. This is a configurable feature, and is described in more detail in section [Section 4.8.4.2](#)
- Any data access that violates this requirement is considered unauthorized and returns an error response.
- All debugger (JTAG or SWD) or bootloader (BSL) accesses to secure memory zones are treated as unauthorized and also return an error response.

An error response generated in the system results in an exception condition to the processor and must be handled as described in the *Cortex-M4F Processor* chapter.

**Figure 4-1** shows the IP protected secure zones within the device when all of the four IP protection secure zones are configured.



**Figure 4-1. IP Protected Secure Zones Representation**

---

**NOTE:** MSP432P4xx devices prevent Instruction Fetches to the SBUS space when security features (JTAG and SWD lock and/or IP protection) are enabled. This is to ensure that IP protection features are not compromised through the flash patching feature of the Cortex®-M4. When device security is enabled, any instruction fetch on SBUS space results in the SYSCTL initiating a reboot of the device.

---

#### 4.8.4.1 Execution of IP Protected Secure Zone code

Protected execution deals with the case when the device implements IP protection and a secure function is called. During the execution of the secure code, if the debugger connection is kept active, attempts could be made to halt the processor and reverse engineer code contents by accessing the CPU registers (with single-step iterations). This is prevented by disabling the debugger connection whenever execution is inside a secure zone.

If and when the CPU execution enters an IP protected secure memory zone, the security control mechanism alerts the SYSCTL module, after which all JTAG/SWD (debugger) accesses into the CPU's debug AHB-AP port are disabled. In addition, the SYSCTL prevents the CPU from halting inside a secure zone to prevent a possible security hole, or a device lock-up condition (halted device unable to restart). This feature also enables code within secure zones to use the device's internal SRAM for temporary storage of critical/proprietary data. As long as the CPU is executing from within the secure zone, the full device (and hence the SRAM) is inaccessible to unauthorized external mechanisms. It is the responsibility of the secure code to ensure that all critical data are erased from the SRAM and peripheral registers before control is handed back to non-secure zones (thus unlocking the device to subsequent debugger accesses).

The SYSCTL security control mechanism monitors accesses from the debugger in the DAP (debug) port of the CPU and selectively allows, filters, or blocks those accesses if Debug security is active. Debug security is typically active in one of two conditions:

- The device is in JTAG and SWD lock mode.
- The device has IP protection enabled, and the CPU is currently executing in one of the defined secure memory zones

Under Debug security active condition, SYSCTL also does not permit the CPU to halt. Ideally, preventing any debugger accesses should also prevent halt conditions, but malicious software may be able to enable breakpoint addresses that point into secure memory zones and, thereby cause a halt when the CPU is executing secure code. In this case, the hardware ensures that the CPU halt condition is not honored and the CPU is not allowed to halt.

When Debug security is inactive, all debugger accesses are allowed to pass through to the DAP port of the CPU.

#### 4.8.4.2 IP Protection and Secure Zone Data Access Unlock Register

To provide an extra layer of security for sensitive data content in memory, the IP protection control can be configured to enable the Data Access Locking feature of the device security infrastructure. If Data Access Locking is enabled, even secure code cannot access data from within its own secure zone. This Data access locking is achieved through the SEC\_ZONE<sub>x</sub>\_DATA\_EN fields in the flash boot override mailbox (FL\_BOOTOVER\_MAILBOX). See [Section 4.8.6](#) for details on boot overrides.

If the Data Access Lock is disabled, the secure code is permitted to access data from within its own secure zone. However, this needs an additional step whereby secure code first needs to explicitly request unlock of data accesses to its own zone. This unlocking is carried out through the Secure Zone Data Unlock Register (SYS\_SECDATA\_UNLOCK). Unlock commands to the Secure Zone Data Unlock Register are honored only if the following conditions are satisfied:

- The IP protected secure zone data enable (SEC\_ZONE<sub>x</sub>\_DATA\_EN) field in the boot-override mailbox was set to enable (0x00000000) when the secure zone was set up.
- The code writing-to the SYS\_SECDATA\_UNLOCK register (and thereby requesting data accesses to a secure zone) lies within the same secure zone.
- Writes to the data unlock register use the appropriate unlock key as defined in the SYS\_SECDATA\_UNLOCK register bit description. See [Section 4.11.10](#) for details on the bit description.

When a secure zone is unlocked for data, data accesses to that zone are permitted, but only for code executing from within the same zone.

#### 4.8.5 In-Field Updates

This section primarily deals with in-field updates for firmware or data on MSP432P4xx family of devices for an unsecured/JTAG and SWD locked/IP protected device.

##### 4.8.5.1 In-Field Updates: Unsecure Device

If there is no security enabled on the device (JTAG and SWD lock or IP protection), update to the firmware or data on the device can be done either through the JTAG debugger, [TI Bootloader](#) (BSL), or an application-specific BSL.

##### 4.8.5.2 In-Field Updates: Secure Device

As described in [Section 4.8](#), MSP432P4xx devices support two types of device security:

1. JTAG and SWD Lock
2. IP Protection

This section talks about how in-field updates to firmware or data can be done for a Secure Device

##### 4.8.5.3 In-Field Updates: Secure Device With JTAG and SWD Lock Enabled

Firmware or data load to a JTAG and SWD locked device is done by invoking the BSL. The update could be either of the two following categories:

1. Unencrypted update
2. Encrypted update

#### 4.8.5.3.1 Unencrypted Update: JTAG and SWD Locked Device

Unencrypted update of device firmware or data can be done by invoking the BSL and then doing a data or code download through the BSL into the required address. This is similar to the case of the unsecure device update except that an unencrypted update does not work for data or code being updated into an IP protected secure zone flash memory. For firmware or data updates to IP protected secure zones, see [Section 4.8.5.4](#).

#### 4.8.5.3.2 Encrypted Update: JTAG and SWD Locked Device

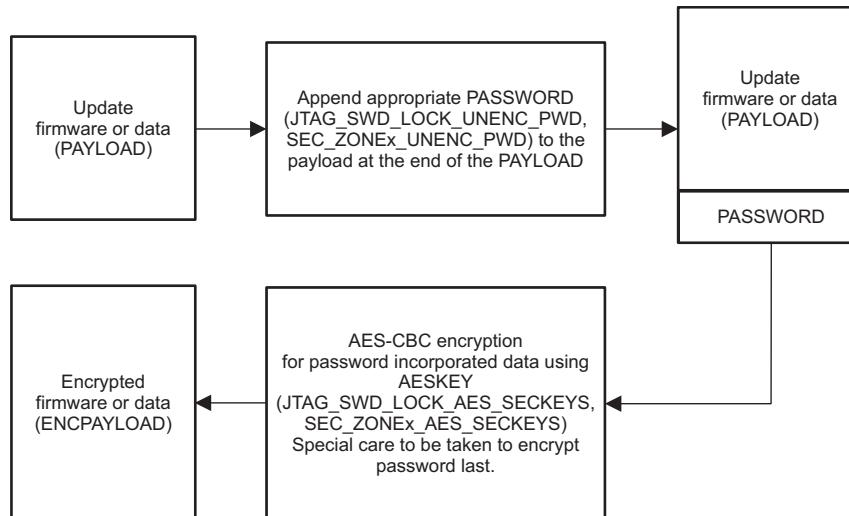
Firmware or data update can be done to the device by taking advantage of the Encrypted Update boot override mode. This provides an option to encrypt the data at source before the transmitting it to the device.

A data setup phase is needed before using the encrypted update. This should be done by the host system which intends to initiate the firmware or data update to MSP432P4xx devices (see [Figure 4-2](#)). The password shown here should be the same as the password used when enabling the JTAG and SWD lock in the boot override mailbox (JTAG\_SWD\_LOCK\_UNENC\_PWD). This password is first appended to the end of the firmware or data to be updated and then taken through an AES-CBC encryption. The encrypted payload is now transmitted through BSL into a free space in Bank 1 of the device main flash memory.

This is followed by a boot override into the design with the command JTAG\_SWD\_LOCK\_ENC\_UPDATE. See [Section 4.8.6](#) for details on boot-overrides.

The device boot code now finds a boot override command. Boot code decrypts the encrypted packet and checks for the authenticity by comparing the JTAG\_SWD\_LOCK\_UNENC\_PWD from the decrypted packet against the value that were provided during JTAG and SWD lock setup and then performs the update if the passwords match. The status of the update is indicated in the ACK field of the specific command in the boot override mailbox.

The JTAG and SWD Lock parameters: JTAG\_SWD\_LOCK\_AES\_INIT\_VECT, JTAG\_SWD\_LOCK\_AES\_SECKEYS and JTAG\_SWD\_LOCK\_UNENC\_PWD act as master set of parameters for the device. Hence, they can also be used to perform a data or firmware update to the device IP protected regions along with the other flash main memory regions of the device.



**Figure 4-2. Data Setup for Encrypted Update**

#### 4.8.5.4 In-Field Updates: Secure Device With IP Protection Enabled

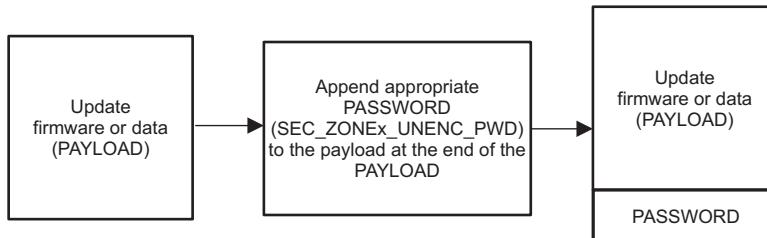
Firmware or data load to an IP protected secure zone is done by invoking the BSL and then subsequently invoking the boot-override mode of the device. The update could be any of the two following categories:

1. Unencrypted update
2. Encrypted update

The user can choose between an encrypted update (AES256-CBC encrypted + 128-bit password authenticated) or unencrypted update (authenticated by a 128-bit password) for a particular IP protected secure zone. The use of an encrypted or unencrypted update is controlled by the user securing the IP protected zone at the time of IP protection setup. IP protected secure zones configured with unencrypted updates can be updated using unencrypted update only. Similarly, IP protected secure zones configured with encrypted updates can be updated using the Encrypted update only.

#### 4.8.5.4.1 Unencrypted Update: IP Protected Device

The unencrypted update of an IP protected secure zone uses a password-based mechanism to ensure that a valid user is trying to update a secure area. To use this mode, first append the 128-bit password to the end of the data to be updated (see [Figure 4-3](#)).



**Figure 4-3. Data Setup for IP Protected Secure Zone Unencrypted Update**

The password used here should be the same as SEC\_ZONE<sub>x</sub>\_UNENC\_PWD provided when setting up the IP protected secure zone and is different from the BSL password. The payload is now transmitted through BSL into a free space in Bank 1 of the device main flash memory.

This is followed by a boot override into the design with the command SEC\_ZONE<sub>x</sub>\_UPDATE. See [Section 4.8.6](#) for details on boot-overrides.

The device boot code now finds a boot override command. Boot code checks for the authenticity by comparing the SEC\_ZONE<sub>x</sub>\_UNENC\_PWD from the payload against the values that were provided during IP Protected secure zone setup and then performs the update if the passwords match. The status of the update is indicated in the ACK field of the specific command in the boot override mailbox.

#### 4.8.5.4.2 Encrypted Update: IP Protected Device

The encrypted update of an IP protected secure zone is similar to the encrypted update of a device with JTAG and SWD lock enabled.

A data setup phase is needed prior to use of the Encrypted update. This is shown in [Figure 4-2](#). The password shown here should be the same as the password used when enabling the IP Protected secure zone in the boot override mailbox (SEC\_ZONE<sub>x</sub>\_UNENC\_PWD). This password is first appended to the end of the firmware or data to be updated and then taken through an AES-CBC encryption. The encrypted payload is now transmitted through BSL into a free space in Bank 1 of the device main flash memory.

This is followed by a boot override into the design with the command SEC\_ZONE<sub>x</sub>\_UPDATE. See [Section 4.8.6](#) for details on boot overrides.

The device boot code now finds a boot override command. Boot code decrypts the encrypted packet and checks for the authenticity by comparing the SEC\_ZONE<sub>x</sub>\_UNENC\_PWD from the decrypted packet against the value that were provided during IP Protected secure zone setup and then performs the update if the passwords match. The status of the update is indicated in the ACK field of the specific command in the boot override mailbox.

### 4.8.6 Boot-Overrides

Applications running on MSP432P4xx devices can initiate boot-overrides into the system. Boot-overrides are special boot modes in the system, where application can send a command to the device boot-code. The following are the main uses of boot-overrides:

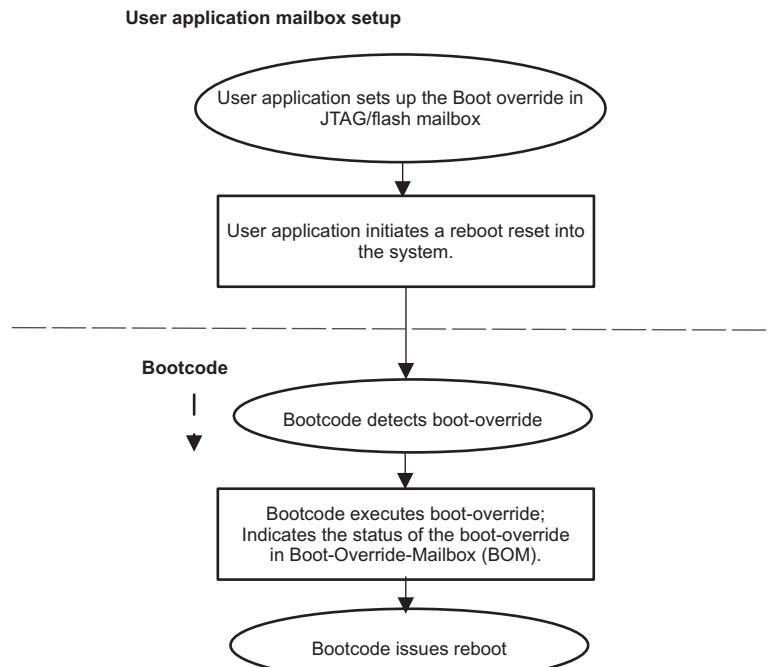
- Setting up device JTAG and SWD lock.

- Setting up device IP protection.
- Factory reset the device to remove all security definitions and erase flash main memory.
- Setting up factory reset configurations in the device.
- Setting up BSL configurations in the device.

Boot overrides can be achieved in the system using either of the approaches below:

- Boot-override through JTAG and SWD
- Boot-override through flash mailbox

[Figure 4-4](#) shows the general boot-override flow for JTAG Flash mailbox in the MSP432P4xx devices.



**Figure 4-4. Boot Override Flow**

#### 4.8.6.1 Boot Override Through JTAG

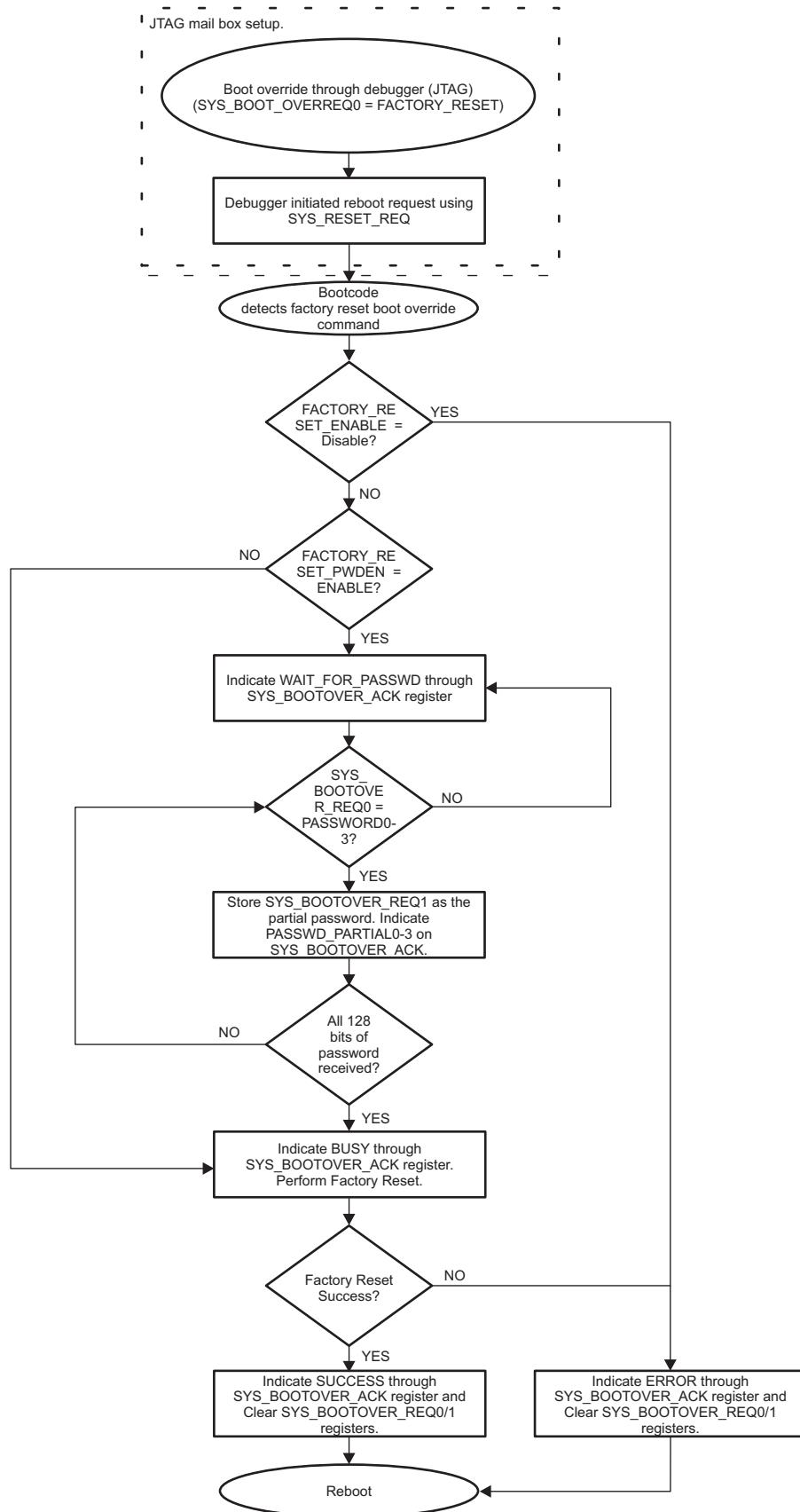
The SYSCTL module supports only one boot-override command over JTAG:

- Factory Reset: To erase the entire flash main memory. Removes all security definitions in the system.

The SYSCTL module implements a register level infrastructure for boot-overrides. These registers are reset only on a POR class of reset and hence can be effectively used to communicate between the application and the boot-code present within the system.

Boot override over JTAG uses the SYS\_BOOTOVER\_REQ0, SYS\_BOOTOVER\_REQ1 and the SYS\_BOOTOVER\_ACK registers to communicate to the boot-code and initiate these operations. SYS\_BOOTOVER\_REQ0 is used as the command input register for the boot-override whereas SYS\_BOOTOVER\_REQ1 is used as a parameter input register for the command. SYS\_BOOTOVER\_ACK reflects the success/fail status from the bootcode for the command.

To enable the boot-override modes of operation, the user needs to program the SYS\_BOOTOVER\_REQ0 register with the command and the SYS\_BOOTOVER\_REQ1 register with the parameter for the command. When this is done, the application must initiate a REBOOT using the SYS\_RESET\_REQ register to start the boot process. The boot-code now finds a boot-override request command on the SYS\_BOOTOVER\_REQ0 register and proceeds to execute the required command. [Figure 4-5](#) provides the flow-chart for invoking factory reset boot override command through JTAG.


**Figure 4-5. Factory Reset Boot Override Command Through JTAG**

---

**NOTE:** The command for factory reset boot override through JTAG is lost on a device power cycle. So care should be taken to ensure that power cycle does not happen during this operation. In the event of a power cycle, the factory reset command through JTAG should be re-initiated.

---

#### 4.8.6.2 Boot Override Through Flash Mailbox

Boot overrides through flash mailbox are used for the following functionality in MSP432P4xx devices:

- JTAG and SWD lock enable: To enable the JTAG and SWD lock for the device.
- IP protection enable: To enable the IP protection definition for any combination of the four different secure zones needed by the application.
- Factory Reset: To erase the entire flash main memory. Remove all security definitions in the system.
- Factory Reset configuration: To enable password for factory reset or to disable factory reset functionality.
- BSL configuration: To enable or disable the device BSL and to set up various BSL parameters
- In-field update: Encrypted update for JTAG and SWD locked or encrypted or unencrypted update to any of the IP protected secure zone in the device

The security settings of the MSP432P4xx devices can be configured by the boot-code on the basis of a flash based mail box, FL\_BOOTOVER\_MAILBOX. The flash mailbox is located in the information memory of flash. See the device datasheet for the actual address of the flash mailbox. The flash mailbox defines a series of commands and parameters which have to be setup by the user. The flash mailbox can be programmed with the appropriate values like any other flash location. When the setup of the flash mailbox is complete, the application must initiate a REBOOT using the SYS\_RESET\_REQ register. The boot-code now finds a boot-override request command in the flash mailbox and proceeds to execute the required command.

##### 4.8.6.2.1 Boot Override Flash Mailbox (FL\_BOOTOVER\_MAILBOX)

The structure of the mailbox is given below. Only relative addresses have been provided for the flash mailbox. Absolute addresses may be device dependent, and this data is available on the device-specific data sheet.

**Table 4-1. Boot Override Flash Mailbox**

Mailbox Offset	Group	Description	Value
0x0		MB_START	Mail box start (must be programmed by the user for the boot code to determine a valid flash boot override mailbox) 0x0115ACF6
0x4	GEN_PARAMS	CMD	Command for boot override operations. To be programmed by user when setting up the flash mailbox.
0x8	Reserved	Reserved	0xFFFFFFFF
0xC		Reserved	0xFFFFFFFF
0x10	JTAG_SWD_LOCK_PARAMS	JTAG_SWD_LOCK_SECEN	JTAG and SWD lock enable Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x14-0x20		JTAG_SWD_LOCK_AES_INIT_VECT[0-3]	JTAG and SWD lock AES initialization vector for AES-CBC to be used for encrypted updates
0x24-0x40		JTAG_SWD_LOCK_AES_SECKEYS[0-7]	JTAG and SWD lock AES CBC security keys 0-7. This is the key that is used to generate the ENCPAYLOAD when the user intends to do an upgrade. 0xFFFFFFFF when security is disabled.
0x44-0x50		JTAG_SWD_LOCK_UNENC_PWD[0-3]	JTAG and SWD lock unencrypted password 0xFFFFFFFF when security is disabled.
0x54		ACK	Acknowledgment for this command
0x58-0x5C		Reserved	Reserved

**Table 4-1. Boot Override Flash Mailbox (continued)**

Mailbox Offset	Group	Description	Value
0x60	SEC_ZONE0_PARAMS	SEC_ZONE0_SECEN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x64		SEC_ZONE0_START_ADDR	Start address of IP protected secure zone 0. Should be in Bank 0 and aligned to 4KB boundary in main flash region.
0x68		SEC_ZONE0_LENGTH	Length of IP protected secure zone 0 in bytes. Should be multiples of 4KB flash sector size.
0x6C-0x78		SEC_ZONE0_AESINIT_VECT[0-3]	IP protected secure zone 0 AES initialization vector for AES-CBC to be used for encrypted updates.
0x7C-0x98		SEC_ZONE0_SECKEYS[0-7]	AES-CBC security keys. This should be the key that is used to generate the ENCPAYLOAD when the user intends to do an upgrade. 0xFFFFFFFF when IP protected secure zone 0 security disabled.
0x9C-0xA8		SEC_ZONE0_UNENC_PWD[0-3]	Unencrypted password for authentication. 0xFFFFFFFF when IP protected secure zone 0 security disabled.
0xAC		SEC_ZONE0_ENCUPDATE_EN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0xB0		SEC_ZONE0_DATA_EN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0xB4		ACK	Acknowledgment for this command
0xB8-BC		RESERVED	0xFFFFFFFF
0xC0	SEC_ZONE1_PARAMS	SEC_ZONE1_SECEN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0xC4		SEC_ZONE1_START_ADDR	Start address of IP protected secure zone 1. Should be in Bank 0 and aligned to 4KB boundary in main flash region.
0xC8		SEC_ZONE1_LENGTH	Length of IP protected secure zone 1 in bytes. Should be multiples of 4KB flash sector size.
0xCC-0xD8		SEC_ZONE1_AESINIT_VECT[0-3]	IP protected secure zone 1 AES initialization vector for AES-CBC to be used for Encrypted Updates.
0xDC-0xF8		SEC_ZONE1_SECKEYS[0-7]	AES-CBC security keys. This should be the key that is used to generate the ENCPAYLOAD when the user intends to do an upgrade. 0xFFFFFFFF when IP protected secure zone 1 security disabled.
0xFC-0x108		SEC_ZONE1_UNENC_PWD[0-3]	Unencrypted password for authentication. 0xFFFFFFFF when IP protected secure zone 1 security disabled.
0x10C		SEC_ZONE1_ENCUPDATE_EN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x110		SEC_ZONE1_DATA_EN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x114		ACK	Acknowledgment for this command
0x118-0x11C		RESERVED	0xFFFFFFFF

**Table 4-1. Boot Override Flash Mailbox (continued)**

Mailbox Offset	Group	Description	Value
0x120	SEC_ZONE2_PARAMS	SEC_ZONE2_SECEN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x124		SEC_ZONE2_START_ADDR	Start address of IP protected secure zone 2. Should be in Bank 0 and aligned to 4KB boundary in main flash region.
0x128		SEC_ZONE2_LENGTH	Length of IP protected secure zone 2 in bytes. Should be multiples of 4KB flash sector size.
0x12C-0x138		SEC_ZONE2_AESINIT_VECT[0-3]	IP protected secure zone 2 AES initialization vector for AES-CBC to be used for Encrypted Updates.
0x13C-0x158		SEC_ZONE2_SECKEYS[0-7]	AES-CBC security keys. This should be the key that is used to generate the ENCPAYLOAD when the user intends to do an upgrade. 0xFFFFFFFF when IP protected secure zone 2 security disabled.
0x15C-0x168		SEC_ZONE2_UNENC_PWD[0-3]	Unencrypted password for authentication. 0xFFFFFFFF when IP protected secure zone 2 security disabled.
0x16C		SEC_ZONE2_ENCUPDATE_EN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x170		SEC_ZONE2_DATA_EN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x174		ACK	Acknowledgment for this command
0x178-0x17C		RESERVED	0xFFFFFFFF
0x180	SEC_ZONE3_PARAMS	SEC_ZONE3_SECEN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x184		SEC_ZONE3_START_ADDR	Start address of IP protected secure zone 3. Should be in Bank 0 and aligned to 4KB boundary in main flash region.
0x188		SEC_ZONE3_LENGTH	Length of IP protected secure zone 3 in bytes. Should be multiples of 4KB flash sector size.
0x18C-0x198		SEC_ZONE3_AESINIT_VECT[0-3]	IP protected secure zone 3 AES initialization vector for AES-CBC to be used for Encrypted Updates.
0x19C-0x1B8		SEC_ZONE3_SECKEYS[0-7]	AES-CBC security keys. This should be the key that is used to generate the ENCPAYLOAD when the user intends to do an upgrade. 0xFFFFFFFF when IP protected secure zone 3 security disabled.
0x1BC-0x1C8		SEC_ZONE3_UNENC_PWD[0-3]	Unencrypted password for authentication. 0xFFFFFFFF when IP protected secure zone 3 security disabled.
0x1CC		SEC_ZONE3_ENCUPDATE_EN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x1D0		SEC_ZONE3_DATA_EN	Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x1D4		ACK	Acknowledgment for this command
0x1D8-0x1DC		RESERVED	0xFFFFFFFF

**Table 4-1. Boot Override Flash Mailbox (continued)**

Mailbox Offset	Group	Description	Value
0x1E0	BSL_PARAMS	BSL Enable	Disable = 0xFFFFFFFF Enable = 0x00000000 (any value other than 0xFFFFFFFF) (default state)
0x1E4		BSL Start Address	Contains the pointer to the BSL function. Bootcode reads this location and uses this as a function pointer. Default = TI BSL API table address.
0x1E8		BSL hardware invoke parameters	Hardware invoke field. boot code jumps to the BSL after matching the values in the field with the corresponding port pins of the device.
			Bit 31      1h = Disable 0h = Enable
			Bits 30:26    Reserved. Default should be 0x1F.
			Bits 25:16    I <sup>2</sup> C slave address. Default = 0x48.
			Bits 15:13    Interface selection. 7h = Automatic 6h = UART 5h = SPI 4h = I <sup>2</sup> C 3h-0h = Reserved for future expansion; Defaults to automatic mode.
			Bit 12      Polarity of port pin. 0h = Low 1h = High
			Bits 11:7    Reserved for future expansion. Default should be 0x1F.
			Bits 6:4      Pin number of the port to be used for BSL entry. 0h = BIT0 1h = BIT1 2h = BIT2 3h = BIT3 4h = BIT4 5h = BIT5 6h = BIT6 7h = BIT7
			Bits 3:0      Port to be used for HW BSL entry sequence. 0h = P1 1h = P2 2h = P3 3h-Fh = Reserved
0x1EC-0x1F0		Reserved	0xFFFFFFFF
0x1F4		ACK	Acknowledgment for this command
0x1F8	JTAG_SWD_LOCK_ENC_UPDATE	JTAG_SWD_LOCK_ENCPAYLOADADDR	Start address where the payload is loaded in the device. Should be in Bank 1 of the main flash memory.
0x1FC		JTAG_SWD_LOCK_ENCPAYLOADLEN	Length of the encrypted payload in bytes + 128 bits of password. This value should be a multiple of 4KB flash sector size + 128 bit password.
0x200		JTAG_SWD_LOCK_DST_ADDR	Destination address where the final data needs to be stored into the device. Should be aligned to 4KB boundary.
0x204		ACK	Acknowledgment for this command
0x208		RESERVED	0xFFFFFFFF
0x20C	SEC_ZONE0_UPDATE	SEC_ZONE0_ENCPAYLOADADDR	Start address where the payload is loaded in the device. Should be in Bank 1 of the main flash memory.
0x210		SEC_ZONE0_ENCPAYLOADLEN	Length of the payload in bytes. This value should match the SEC_ZONE0_LENGTH parameter + 128 bits. There is a limitation that the IP protected secure zone update restricts the user to update a full secure zone. User cannot initiate a partial update to the IP protected secure zone.
0x214		ACK	Acknowledgment for this command
0x218		Reserved	0xFFFFFFFF

**Table 4-1. Boot Override Flash Mailbox (continued)**

Mailbox Offset	Group	Description	Value
0x21C	SEC_ZONE1_UPDATE	SEC_ZONE1_ENCPAYLOADADDR	Start address where the payload is loaded in the device. Should be in Bank 1 of the main flash memory.
0x220		SEC_ZONE1_ENCPAYLOADLEN	Length of the payload in bytes. This value should match the SEC_ZONE1_LENGTH parameter + 128 bits. There is a limitation that the IP protected secure zone update restricts the user to update a full secure zone. User cannot initiate a partial update to the IP protected secure zone.
0x224		ACK	Acknowledgment for this command
0x228		Reserved	0xFFFFFFFF
0x22C	SEC_ZONE2_UPDATE	SEC_ZONE2_ENCPAYLOADADDR	Start address where the payload is loaded in the device. Should be in Bank 1 of the main flash memory.
0x230		SEC_ZONE2_ENCPAYLOADLEN	Length of the payload in bytes. This value should match the SEC_ZONE2_LENGTH parameter + 128 bits. There is a limitation that the IP protected secure zone update restricts the user to update a full secure zone. User cannot initiate a partial update to the IP protected secure zone.
0x234		ACK	Acknowledgment for this command
0x238		Reserved	0xFFFFFFFF
0x23C	SEC_ZONE3_UPDATE	SEC_ZONE3_ENCPAYLOADADDR	Start address where the payload is loaded in the device. Should be in Bank 1 of the main flash memory.
0x240		SEC_ZONE3_ENCPAYLOADLEN	Length of the payload in bytes. This value should match the SEC_ZONE3_LENGTH parameter + 128 bits. There is a limitation that the IP protected secure zone update restricts the user to update a full secure zone. User cannot initiate a partial update to the IP protected secure zone.
0x244		ACK	Acknowledgment for this command
0x248 - 0x24C		Reserved	0xFFFFFFFF
0x250	FACTORY_RESET_PARAMS	FACTORY_RESET_ENABLE	Enable or disable factory reset Enable = 0xFFFFFFFF (default state) Disable = 0x00000000 (any value other than 0xFFFFFFFF)
0x254		FACTORY_RESET_PWDEN	Factory reset password enable Disable = 0xFFFFFFFF (default state) Enable = 0x00000000 (any value other than 0xFFFFFFFF)
0x258 - 0x264		FACTORY_RESET_PWD[0-3]	128-bit password for factory reset to be saved into the device. This field has an effect only if the FACTORY_RESET_ENABLE field is in the "Enable" state.
0x268		ACK	Acknowledgment for this command
0x26C		Reserved	0xFFFFFFFF
0x270 - 0x27C	FACTORY_RESET	PASSWORD[0-3]	128-bit password for factory reset. Has an effect only if the FACTORY_RESET_ENABLE is in "Enable" state and FACTORY_RESET_PWDEN has been set with a password. This field is compared with the password provided with the FACTORY_RESET_PARAMS command before the device is erased.
0x280		ACK	Acknowledgment for this command
0x284 - 0x288		Reserved	Reserved
0x28C		MB_END	Mailbox end (Has to be programmed by the user for the boot code to determine a valid flash boot override mailbox) 0x0011E11D

**NOTE:** The default value of FACTORY\_RESET\_ENABLE is 0xFFFFFFFF which is opposite than the "Enable" setting for all other applicable parameters in the boot override mailbox.

#### 4.8.6.2.2 Boot Override Commands and Acknowledgments

Table 4-1 shows the different CMD and ACK values to be provided by the application to invoke a boot-override.

**Table 4-2. Commands Used by Boot-Code for Boot Override**

CMD	Value
<b>JTAG Based Override Mode</b>	
FACTORY_RESET	0x00000001
PASSWORD3	0x00000800
PASSWORD2	0x00000400
PASSWORD1	0x00000200
PASSWORD0	0x00000100
<b>Flash Mailbox Based Override Mode</b>	
FACTORY_RESET	0x00010000
BSL_CONFIG	0x00020000
JTAG_SWD_LOCK_SECEN	0x00080000
SEC_ZONE0_EN	0x00100000
SEC_ZONE1_EN	0x00200000
SEC_ZONE2_EN	0x00400000
SEC_ZONE3_EN	0x00800000
SEC_ZONE0_UPDATE	0x01000000
SEC_ZONE1_UPDATE	0x02000000
SEC_ZONE2_UPDATE	0x04000000
SEC_ZONE3_UPDATE	0x08000000
JTAG_SWD_LOCK_ENC_UPDATE	0x10000000
FACTORY_RESET_PARAMS	0x20000000
ANY_CONFIG	Set the bits corresponding to any of the previous commands. For example, if you want to enable JTAG_SWD_LOCK_SECEN and SEC_ZONE1_EN then the Command to provide is: 0x00080000   0x00200000 = 0x00280000
NONE	0x00000000; 0xFFFFFFFF
<b>Both JTAG and Flash Mailbox Based Override Mode</b>	
Others	All other values not defined for the override mode.

**Table 4-3. ACKs Used by Boot-Code to Indicate Status of Boot Override**

ACK	Value
BUSY	0x00000001
SUCCESS	0x00000ACE
WAIT_FOR_PASSWD	0x00000002
PASSWD_PARTIAL0	0x00010004
PASSWD_PARTIAL1	0x00020004
PASSWD_PARTIAL2	0x00040004
PASSWD_PARTIAL3	0x00080004
ERROR (JTAG based command)	0xDEAD0000
ERROR (Mailbox based command)	0x0000DEAD
DEFAULT	0xFFFFFFFF

The following is an example of boot-over ride setting.

Assuming that the user wants to enable the IP protection for secure zone 0 on the device, the application must set up the FL\_BOOTOVER\_MAILBOX with the following structure:

- MB\_START= 0x0115ACF6
- CMD = SEC\_ZONE0\_EN
- SEC\_ZONE0\_PARAMS → ACK = 0xFFFFFFFF
- SEC\_ZONE0\_PARAMS → SEC\_ZONE0\_SECEN = ENABLE
- SEC\_ZONE0\_PARAMS → SEC\_ZONE0\_START\_ADDR = Start Address of the secure zone (should be in Bank 0 of the main flash of the device)
- SEC\_ZONE0\_PARAMS → SEC\_ZONE0\_LENGTH = Length of the secure zone (in steps of 4KB)
- SEC\_ZONE0\_PARAMS → SEC\_ZONE0\_DATA\_EN =
  - DISABLE (if data reads from IP protected secure zone should be completely disabled)
  - ENABLE (if data reads from IP protected secure zone should be permitted based on the unlock conditions in SYS\_SECDATA\_UNLOCK register)
- SEC\_ZONE0\_PARAMS → SEC\_ZONE0\_ENCUPDATE\_EN =
  - ENABLE (if encrypted update to the IP protected zone is desired)
  - DISABLE (if unencrypted update to the IP protected zone is desired)
- All other locations = 0xFFFFFFFF
- MB\_END = 0x0011E11D

The application should then initiate a reboot into the system. The boot-code now sees a boot-override sequence and proceeds to process it. After the execution of the override command, boot code overrides the command programed in the mailbox to NONE and then reboots the device. The device should now be secure with IP protection, if the boot-override ACK field has a value of 0xACE.

#### **4.8.6.2.3 Using ANY\_CONFIG Command**

This command is used to process many of the boot-override commands simultaneously if desired. The boot-code executes each of the enabled commands and indicates an error with the ERROR code if any of them fails.

For example, the user would need to do the following to setup IP protection on secure zone 0 along with JTAG and SWD lock in a single boot override session:

- MB\_START = 0x0115ACF6
- CMD = SEC\_ZONE0\_EN | JTAG\_SWD\_LOCK\_SECEN
- JTAG\_SWD\_LOCK\_PARAMS → JTAG\_SWD\_LOCK\_SECEN = ENABLE
- JTAG\_SWD\_LOCK\_PARAMS → ACK = DEFAULT
- SEC\_ZONE0\_PARAMS → SEC\_ZONE0\_SECEN = ENABLE
- SEC\_ZONE0\_PARAMS → SEC\_ZONE0\_START\_ADDR = Start Address of the secure zone
- SEC\_ZONE0\_PARAMS → SEC\_ZONE0\_LENGTH = Length of the secure zone
- SEC\_ZONE0\_PARAMS → SEC\_ZONE0\_DATA\_EN = ENABLE (if data reads from IP protected secure zone should be permitted based on the unlock conditions in SYS\_SECDATA\_UNLOCK register)
- SEC\_ZONE0\_PARAMS → ACK = 0xFFFFFFFF
- All other locations = 0xFFFFFFFF
- MB\_END = 0x0011E11D

The application should then initiate a reboot into the system. The boot-code now sees two boot-override sequences and proceeds to process one after the other. After the execution of the override commands, boot code overrides the commands programed in the mailbox to NONE and then reboots the device. The device should now be secured with both IP protection and JTAG and SWD lock, if the boot-override ACK field has a value of 0xACE.

#### 4.8.7 Device Security and Boot Overrides User Considerations

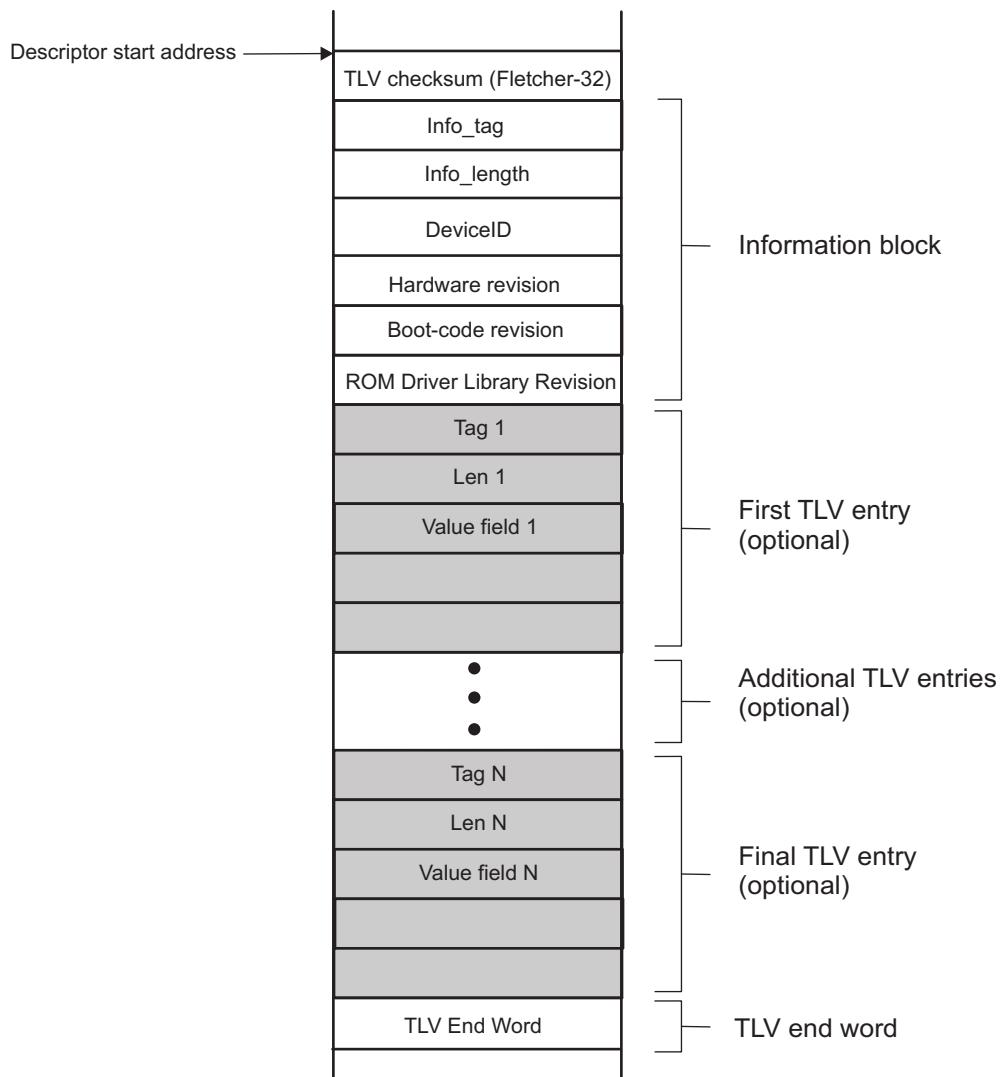
This section provides some guidelines to users for using device security and boot overrides.

1. After any of the encrypted in-field updates, user should delete the ENCPAYLOAD that was loaded into the flash memory of the device. The boot-code does not delete this data after the in-field update is complete. However, the boot code cleans up the decrypted data from the device before handing over control to the user.
2. IP Protection must not be enabled for the first sector in flash main memory (address location: 0x0, length: 4KB) if in-field updates and use of TI BSL is desired.
3. IP protection setup is allowed only in Bank 0 of main memory of flash.
4. Hardware invocation of BSL is not supported by default. Users must enable it using the boot override mailbox.
5. The maximum size limit for ENCPAYLOAD is 16 KB less than the total SRAM size while performing encrypted updates to IP protected secure zone or JTAG and SWD locked device. Updates larger than this size limit must use multiple boot-override sequences.
6. If encrypted in-field updates are needed for IP protected zones:
  - (a)  $\text{SEC\_ZONE}_x\text{\_LENGTH} \leq \text{SYS\_SRAM\_SIZE} - 16\text{KB}$ ; where  $x = 0, 1, 2$  or  $3$
  - (b) IP protected zones with size not meeting the criteria mentioned above should be split to multiple IP protected zones.
7. Encrypted updates to the IP protected secure zone or JTAG and SWD locked device should be done in multiples of 4KB (flash sector size). Users should also ensure that the start address of the encrypted update is a 4KB aligned address.
8. When Factory Reset is disabled, it cannot be enabled again.

## 4.9 Device Descriptor Table

Each device provides a data structure in memory that allows unique identification of the device. The validity of the device descriptor can be verified by Fletcher-32 checksum. Refer to [Section 4.9.2](#) for more details.

[Figure 4-6](#) shows the logical order and structure of the device descriptor table. The complete device descriptor table and its contents can be found in the device-specific data sheet.



**Figure 4-6. Device Descriptor Table**

#### 4.9.1 Tag Length Value (TLV) Descriptors

Each TLV descriptor contains a tag field that identifies the descriptor type. [Table 4-4](#) lists the supported tags. Each tag field is unique to its respective descriptor and is always followed by a length field. The length field represents the length of the descriptor in words. See the device-specific data sheet for the complete TLV structure and which descriptors are available.

**Table 4-4. Tag Values**

Short Name	Value	Description
Reserved	00000001h	Reserved for future use
Reserved	00000002h	Reserved for future use
TAG_CS	00000003h	Clock System Calibration Tag
TAG_FLASH	00000004h	Flash Info Tag
TAG_ADC14	00000005h	ADC14 Calibration Tag
Reserved	00000006h	Reserved for future use
Reserved	00000007h	Reserved for future use
TAG_REF	00000008h	REF Calibration Tag
Reserved	00000009h	Reserved for future use
Reserved	0000000Ah	Reserved for future use
TAG_DEVINFO	0000000Bh	Device Info Tag
TAG_DIEREC	0000000Ch	Die Record Tag
TAG_RANDNUM	0000000Dh	Random Number Tag
Reserved	0000000Eh	Reserved for future use
TAG_BSL	0000000Fh	BSL Configuration Tag

#### 4.9.2 TLV Checksum

The Fletcher-32 checksum is used in the TLV. The checksum can be computed with the following code snippet and with a seed value of DADA0000h.

```
//Calculates a Fletcher-32 checksum, which consists of two parts.
//The first is a checksum of the data values. The second
//is a checksum of the intermediate values of the first checksum.

uint32 Calc_Fletcher32_Chksum(uint16 *data_start, uint32 num_data_values, uint32 seed)
{
    uint32 sum1, sum2, c;

    sum1 = seed & 0xFFFF;
    sum2 = (seed & 0xFFFF0000) >> 16;

    for (c = 0; c < num_data_values; c++)
    {
        //Add the new data value, doing an end-around carry to implement a
        //one's complement sum. This is mostly done for better error detection
        //since it makes an MSB flip affect more than just the MSB of the
        //checksum. It's also endian-independent (up to a bit rotation) and
        //provides easy match detection, although those features don't work in
        //a Fletcher checksum. This step can be optimized for speed by
        //accumulating carries for up to 360 sums, but since the function is
        //only used once per run the code size is optimized instead.
        sum1 += data_start[c];
        sum1 = (sum1 >> 16) + (sum1 & 0xFFFF);
        sum2 += sum1;
        sum2 = (sum2 >> 16) + (sum2 & 0xFFFF);

    }

    return (sum2<<16) | sum1;
}
```

#### 4.9.3 Calibration Values

The TLV structure contains calibration values that can be used to improve the measurement capability of various functions. See the device-specific data sheet for the calibration values that are available on a given device.

##### 4.9.3.1 Clock System Calibration

**Table 4-5** lists the clock system calibration data. This descriptor contains information on DCO frequency calibration value and DCO constant (K) value for DCO frequency ranges (DCORSEL) 0 to 4 and 5. This information is provided for internal and external resistor modes of the DCO. The DCO frequency calibration and the DCO constant values are necessary to program the DCO to any desired frequency in the supported frequency range of 1 to 48 MHz. Refer to [Section 5.2.8.3](#) in the *Clock System (CS)* chapter for more details.

**Table 4-5. Clock System Calibration Data**

Clock System Calibration	Tag	00000003h
	Length	00000010h
	Word	DCO IR mode: Frequency calibration for DCORSEL 0 to 4
	Word	DCO IR mode: Frequency calibration for DCORSEL 5
	Word	Reserved
	Word	DCO IR mode: DCO Constant (K) for DCORSEL 0 to 4
	Word	DCO IR mode: DCO Constant (K) for DCORSEL 5
	Word	DCO ER mode: Frequency calibration for DCORSEL 0 to 4
	Word	DCO ER mode: Frequency calibration for DCORSEL 5
	Word	Reserved
	Word	Reserved
	Word	Reserved
	Word	DCO ER mode: DCO Constant (K) for DCORSEL 0 to 4
	Word	DCO ER mode: DCO Constant (K) for DCORSEL 5

#### 4.9.3.2 Temperature Sensor Calibration

The temperature sensor calibration data is available as part of the ADC calibration descriptor (see [Table 4-6](#)).

**Table 4-6. ADC Calibration Data**

ADC Calibration	Tag	00000005h
	Length	00000018h
	Word	Reserved
	Word	CAL_ADC_12T30
	Word	CAL_ADC_12T85
	Word	CAL_ADC_145T30
	Word	CAL_ADC_145T85
	Word	CAL_ADC_25T30
	Word	CAL_ADC_25T85

The temperature sensor is calibrated using the internal voltage references. Each reference voltage (1.2 V, 1.45 V, or 2.5 V) contains a measured value for two temperatures,  $30^{\circ}\text{C} \pm 3^{\circ}\text{C}$  and  $85^{\circ}\text{C} \pm 3^{\circ}\text{C}$ , stored in the TLV structure. The characteristic equation of the temperature sensor voltage, in millivolts is:

$$V_{\text{SENSE}} = TC_{\text{SENSOR}} \times \text{Temp} + V_{\text{SENSOR}}$$

where

- $TC_{\text{SENSOR}}$ , in  $\text{mV}/^{\circ}\text{C}$ , is the temperature coefficient and represents the slope of the equation.
  - $V_{\text{SENSOR}}$ , in mV, represents the y-intercept of the equation.
  - Temp, in  $^{\circ}\text{C}$ , is the temperature of interest.
- (1)

The temperature (Temp,  $^{\circ}\text{C}$ ) can be computed as follows for each of the reference voltages used in the ADC measurement:

$$\begin{aligned}
 \text{Temp} &= (\text{ADC(raw)} - \text{CAL\_ADC\_12T30}) \times \left( \frac{85 - 30}{\text{CAL\_ADC\_12T85} - \text{CAL\_ADC\_12T30}} \right) + 30 \\
 \text{Temp} &= (\text{ADC(raw)} - \text{CAL\_ADC\_145T30}) \times \left( \frac{85 - 30}{\text{CAL\_ADC\_145T85} - \text{CAL\_ADC\_145T30}} \right) + 30 \\
 \text{Temp} &= (\text{ADC(raw)} - \text{CAL\_ADC\_25T30}) \times \left( \frac{85 - 30}{\text{CAL\_ADC\_25T85} - \text{CAL\_ADC\_25T30}} \right) + 30
 \end{aligned} \tag{2}$$

#### 4.9.3.3 Flash Information

**Table 4-7** lists the flash information descriptor. This descriptor contains information on the maximum number of pulses required for successful flash program or erase operation.

**Table 4-7. Flash Information Descriptor**

Flash Info	Tag	00000004h
	Length	00000002h
	Word	Maximum Programming Pulses
	Word	Maximum Erase Pulses

#### 4.9.3.4 Random Number Seed

**Table 4-8** lists the random number seed.

**Table 4-8. Random Number Seed**

Random Number	Tag	0000000Dh
	Length	00000004h
	4 words	128-bit random number seed

The random number stored as a seed for a deterministic random number generator is programmed during test of the device. It is generated on the test system using a cryptographic random number generator.

#### 4.9.3.5 BSL Configuration

**Table 4-9** lists the BSL configuration data. The BSL configuration stores the communication interface selection and corresponding communication interface settings.

**Table 4-9. BSL Configuration Data**

BSL Configuration	Tag	0000000Fh
	Length	00000004h
	Word	Peripheral Interface Selection
	Word	Port Interface Configuration for UART
	Word	Port Interface Configuration for SPI
	Word	Port Interface Configuration for I <sup>2</sup> C

## 4.10 ARM Cortex-M4F ROM Table Based Part Number

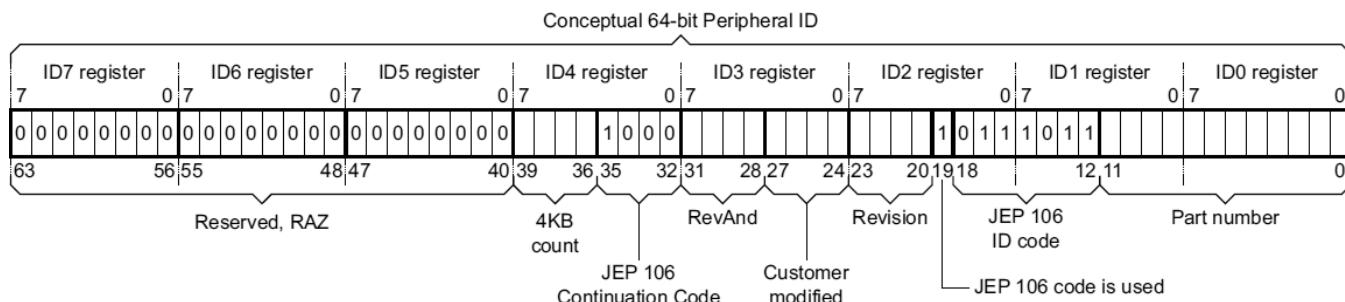
The MSP432P4xx family of MCUs incorporates a part number for the device for the IDEs to recognize the device, in addition to the device IDs specified in the device descriptors (TLV). This section describes how this information is organized on the device.

IEEE 1149.1 defines the use of a IDCODE register in the JTAG chain to provide the fields in [Table 4-10](#)

**Table 4-10. Structure of Device Identification Code**

Bit Position	Field Description
31-28	Version
27-12	Part Number of the device
11-1	Manufacturer Identity
0	Reserved (Always tied to 1)

On MSP432P4xx MCUs, all these fields are implemented on the ARM Cortex-M4 ROM table. The part number can be read by the IDE tools (TI internal or third party) to determine the device. [Figure 4-7](#) shows the Peripheral ID register bit descriptions from the ARM Cortex-M4 specifications. See the *ARM Debug interface V5 Architecture Specification* for bit-level details on the ARM Cortex-M4 Peripheral ID registers.



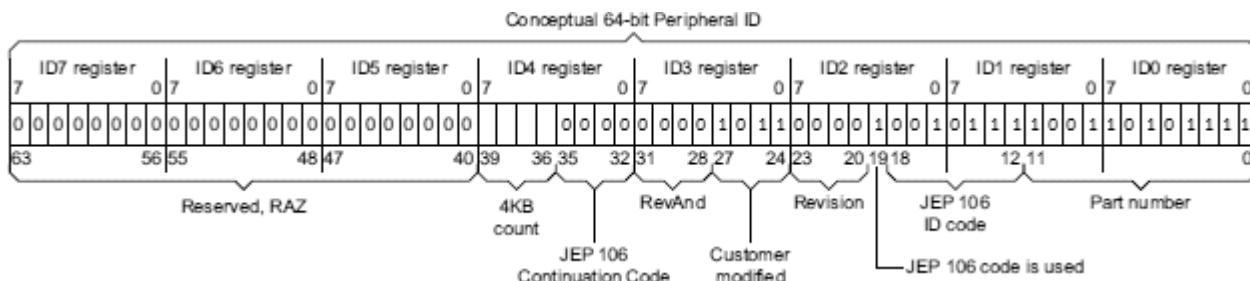
**Figure 4-7. ARM Cortex-M4 Peripheral ID Register Description**

[Figure 4-7](#) shows that a one-to-one mapping is not possible for the following fields from [Table 4-10](#)

1. Version: IEEE 1149.1 defines a 4 bit field where as the Coresight compliant PID registers have 4 bits each for Revision (major revision) and RevAnd (minor revision)
2. Part Number: IEEE 1149.1 defines a 16 bit entity. However, the PID registers in the ROM table have only 12 bits reserved for this purpose (Part number – PID1 and PID0 registers).

For the MSP432P4xx MCUs, the Revision and RevAnd fields are used for tracking the major and minor revisions. Also the Customer modified (4 bit) field is used for extending the Part number to 16 bits, to accommodate all of the fields needed by IEEE 1149.1 in the ROM table.

As an example, the ROM table with IEEE 1149.1-compliant device IDCODE for the MSP432P401xx MCU is 0000-1011-1001-1010-1111-0000-0010-1111 (see [Figure 4-8](#)).



**Figure 4-8. Example of ROM PID Entries for MSP432P401xx MCU**

## 4.11 SYSCTL Registers

This section describes the registers in the SYSCTL module. [Table 4-11](#) shows the registers with their address offsets.

**Table 4-11. SYSCTL Registers**

Offset	Acronym	Register Name	Section
0000h	SYS_REBOOT_CTL	Reboot Control Register	<a href="#">Section 4.11.1</a>
0004h	SYS_NMI_CTLSTAT	NMI Control and Status Register	<a href="#">Section 4.11.2</a>
0008h	SYS_WDTRESET_CTL	Watchdog Reset Control Register	<a href="#">Section 4.11.3</a>
000Ch	SYS_PERIHALT_CTL	Peripheral Halt Control Register	<a href="#">Section 4.11.4</a>
0010h	SYS_SRAM_SIZE	SRAM Size Register	<a href="#">Section 4.11.5</a>
0014h	SYS_SRAM_BANKEN	SRAM Bank Enable Register	<a href="#">Section 4.11.6</a>
0018h	SYS_SRAM_BANKRET	SRAM Bank Retention Control Register	<a href="#">Section 4.11.7</a>
0020h	SYS_FLASH_SIZE	Flash Size Register	<a href="#">Section 4.11.8</a>
0030h	SYS_DIO_GLTFILT_CTL	Digital I/O Glitch Filter Control Register	<a href="#">Section 4.11.9</a>
0040h	SYS_SECDATA_UNLOCK	IP Protected Secure Zone Data Access Unlock Register	<a href="#">Section 4.11.10</a>
1000h	SYS_MASTER_UNLOCK	Master Unlock Register	<a href="#">Section 4.11.11</a>
1004h	SYS_BOOTOVER_REQ0	Boot Override Request 0 Register	<a href="#">Section 4.11.12</a>
1008h	SYS_BOOTOVER_REQ1	Boot Override Request 1 Register	<a href="#">Section 4.11.13</a>
100Ch	SYS_BOOTOVER_ACK	Boot Override Acknowledge Register	<a href="#">Section 4.11.14</a>
1010h	SYS_RESET_REQ	Reset Request Register	<a href="#">Section 4.11.15</a>
1014h	SYS_RESET_STATOVER	Reset Status and Override Register	<a href="#">Section 4.11.16</a>
1020h	SYS_SYSTEM_STAT	System Status Register	<a href="#">Section 4.11.17</a>

---

**NOTE:** Starting from offset 1000h, the boot override request and acknowledge registers are the ONLY registers that are RW by both the CPU and the debugger. All other SYSCTL registers with address offsets higher than 1000h are reserved (R-0) for the CPU and RW for the debugger. **ALL** registers with address offsets higher than 1000h other than SYS\_MASTER\_UNLOCK register are readable and writable only after SYS\_MASTER\_UNLOCK register has been unlocked.

---



---

**NOTE:** This is a 32-bit module and can be accessed ONLY through word (32-bit) access.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

#### 4.11.1 SYS\_REBOOT\_CTL Register (offset = 0000h)

Reboot Control Register

NOTE: Reboot causes the device to re-initialize itself and causes bootcode to execute again.

**Figure 4-9. SYS\_REBOOT\_CTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKEY								Reserved							
w	w	w	w	w	w	w	w	r	r	r	r	r	r	r	w

**Table 4-12. SYS\_REBOOT\_CTL Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Reads return 0h
15-8	WKEY	W	0h	Key to enable writes to bit 0. Bit 0 is written only if WKEY is 69h in the same write cycle.
7-1	Reserved	R	0h	Reserved. Reads return 0h
0	REBOOT	W	0h	Write 1 initiates a Reboot of the device

#### 4.11.2 SYS\_NMI\_CTLSTAT Register (offset = 0004h)

NMI Control and Status Register

**Figure 4-10. SYS\_NMI\_CTLSTAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												PIN_F LG	PCM_ FLG	PSS_F LG	CS_FL G
r	r	r	r	r	r	r	r	r	r	r	r	rw-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PIN_S RC	PCM_ SRC	PSS_ SRC	CS_S RC
r	r	r	r	r	r	r	r	r	r	r	r	rw-0	rw-1	rw-1	rw-1

**Table 4-13. SYS\_NMI\_CTLSTAT Register Description**

Bit	Field	Type	Reset	Description
31-20	Reserved	R	0h	Reserved. Reads return 0h
19	PIN_FLG	RW	0h	0b = Indicates the RSTn/NMI pin was not the source of NMI 1b = Indicates the RSTn/NMI pin was the source of NMI
18	PCM_FLG	R	0h	0b = indicates the PCM interrupt was not the source of NMI 1b = indicates the PCM interrupt was the source of NMI This flag gets auto-cleared when the corresponding source flag in the PCM is cleared
17	PSS_FLG	R	0h	0b = indicates the PSS interrupt was not the source of NMI 1b = indicates the PSS interrupt was the source of NMI This flag gets auto-cleared when the corresponding source flag in the PSS is cleared
16	CS_FLG	R	0h	0b = indicates CS interrupt was not the source of NMI 1b = indicates CS interrupt was the source of NMI This flag gets auto-cleared when the corresponding source flag in the CS is cleared
15-4	Reserved	R	0h	Reserved. Reads return 0h
3	PIN_SRC <sup>(1)(2)</sup>	RW	0h	0b = configures the RSTn/NMI pin as a source of POR Class Reset 1b = configures the RSTn/NMI pin as a source of NMI <b>Note:</b> Setting this bit to 1 prevents the RSTn pin from being used as a reset. An NMI is triggered by the pin only if a negative edge is detected.
2	PCM_SRC	RW	1h	0b = Disables the PCM interrupt as a source of NMI 1b = Enables the PCM interrupt as a source of NMI
1	PSS_SRC	RW	1h	0b = Disables the PSS interrupt as a source of NMI 1b = Enables the PSS interrupt as a source of NMI
0	CS_SRC	RW	1h	0b = Disables CS interrupt as a source of NMI 1b = Enables CS interrupt as a source of NMI

<sup>(1)</sup> When the device enters LPM3/LPM4 modes of operation, the functionality selected by this bit is retained. If selected as an NMI, activity on this pin in LPM3/LPM4 wakes the device and processes the interrupt, without causing a POR. If selected as a Reset, activity on this pin in LPM3/LPM4 causes a device-level POR.

<sup>(2)</sup> When the device enters LPM3.5/LPM4.5 modes of operation, this bit is always cleared to 0. In other words, the RSTn/NMI pin always assumes a reset functionality in LPM3.5/LPM4.5 modes.

#### 4.11.3 SYS\_WDTRESET\_CTL Register (offset = 0008h)

Watchdog Reset Control Register. Controls the class of reset generated by the WDT events.

**Figure 4-11. SYS\_WDTRESET\_CTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw-1	rw-1

**Table 4-14. SYS\_WDTRESET\_CTL Register Description**

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0h	Reserved. Reads return 0h
1	VIOLATION	RW	1h	0b = WDT password violation event generates Soft reset 1b = WDT password violation event generates Hard reset
0	TIMEOUT	RW	1h	0b = WDT timeout event generates Soft reset 1b = WDT timeout event generates Hard reset

#### 4.11.4 SYS\_PERIHALT\_CTL Register (offset = 000Ch)

Peripheral Halt Control Register

**Figure 4-12. SYS\_PERIHALT\_CTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HALT_DMA	HALT_WDT	HALT_ADC	HALT_eUB3	HALT_eUB2	HALT_eUB1	HALT_eUB0	HALT_eUA3	HALT_eUA2	HALT_eUA1	HALT_eUA0	HALT_T32_0	HALT_T16_3	HALT_T16_2	HALT_T16_1	HALT_T16_0
rw-0	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 4-15. SYS\_PERIHALT\_CTL Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Reads return 0h
15	HALT_DMA	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
14	HALT_WDT	RW	1h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
13	HALT_ADC	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
12	HALT_eUB3	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
11	HALT_eUB2	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
10	HALT_eUB1	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
9	HALT_eUB0	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
8	HALT_eUA3	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
7	HALT_eUA2	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
6	HALT_eUA1	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
5	HALT_eUA0	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
4	HALT_T32_0	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
3	HALT_T16_3	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
2	HALT_T16_2	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
1	HALT_T16_1	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted
0	HALT_T16_0	RW	0h	0b = peripheral operation unaffected when CPU is halted 1b = freezes peripheral operation when CPU is halted

#### 4.11.5 SYS\_SRAM\_SIZE Register (offset = 0010h)

SRAM Size Register

**Figure 4-13. SYS\_SRAM\_SIZE Register**

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

**Table 4-16. SYS\_SRAM\_SIZE Register Description**

Bit	Field	Type	Reset	Description
31-0	SIZE	R	variable	Indicates the size of SRAM on the device. See the device-specific data sheet for the amount of SRAM available for the device.

#### 4.11.6 SYS\_SRAM\_BANKEN Register (offset = 0014h)

SRAM Bank Enable Register

**Figure 4-14. SYS\_SRAM\_BANKEN Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															SRAM_RDY
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															BNK7_EN BNK6_EN BNK5_EN BNK4_EN BNK3_EN BNK2_EN BNK1_EN BNKO_EN
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r-1

**Table 4-17. SYS\_SRAM\_BANKEN Register Description**

Bit	Field	Type	Reset	Description
31-17	Reserved	R	0h	Reserved. Reads return 0h
16	SRAM_RDY <sup>(1)</sup>	R	0h	1b = SRAM is ready for accesses. All SRAM banks are enabled/disabled according to values of bits 7:0 of this register 0b = SRAM is not ready for accesses. Banks are undergoing an enable or disable sequence, and reads or writes to SRAM are stalled until the banks are ready.
15-8	Reserved	R	0h	Reserved. Reads return 0h
7	BNK7_EN <sup>(2)</sup>	RW	1h	0b = Disables Bank7 of the SRAM 1b = enables Bank7 of the SRAM When set to 1, bank enable bits for all banks below this bank are set to 1 as well.
6	BNK6_EN <sup>(2)</sup>	RW	1h	0b = Disables Bank6 of the SRAM 1b = enables Bank6 of the SRAM When set to 1, bank enable bits for all banks below this bank are set to 1 as well.
5	BNK5_EN <sup>(2)</sup>	RW	1h	0b = Disables Bank5 of the SRAM 1b = enables Bank5 of the SRAM When set to 1, bank enable bits for all banks below this bank are set to 1 as well.
4	BNK4_EN <sup>(2)</sup>	RW	1h	0b = Disables Bank4 of the SRAM 1b = enables Bank4 of the SRAM When set to 1, bank enable bits for all banks below this bank are set to 1 as well.
3	BNK3_EN <sup>(2)</sup>	RW	1h	0b = Disables Bank3 of the SRAM 1b = enables Bank3 of the SRAM When set to 1, bank enable bits for all banks below this bank are set to 1 as well.
2	BNK2_EN <sup>(2)</sup>	RW	1h	0b = Disables Bank2 of the SRAM 1b = enables Bank2 of the SRAM When set to 1, bank enable bits for all banks below this bank are set to 1 as well.
1	BNK1_EN <sup>(2)</sup>	RW	1h	0b = Disables Bank1 of the SRAM 1b = enables Bank1 of the SRAM When set to 1, bank enable bits for all banks below this bank are set to 1 as well.
0	BNKO_EN	R	1h	When 1, enables Bank0 of the SRAM

<sup>(1)</sup> This bit is automatically set to 0 whenever any of the Bank Enable bits in this register are changed, which in turn triggers a power up or power down of the impacted SRAM blocks. It is set to 1 again after the power sequence is complete and the SRAM blocks are ready for subsequent read/write accesses

<sup>(2)</sup> Writes to this bit are allowed ONLY when the SRAM\_RDY bit is set to 1. If the bit is 0, it indicates that the SRAM banks are not ready, and writes to this bit are ignored.

#### 4.11.7 SYS\_SRAM\_BANKRET Register (offset = 0018h)

SRAM Bank Retention Control Register

**Figure 4-15. SYS\_SRAM\_BANKRET Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															SRAM _RDY
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															BNK7_ RET
r	r	r	r	r	r	r	r	rw-1	BNK0_ RET						

**Table 4-18. SYS\_SRAM\_BANKRET Register Description**

Bit	Field	Type	Reset	Description
31-17	Reserved	R	0h	Reserved. Reads return 0h
16	SRAM_RDY <sup>(1)</sup>	R	0h	1b = SRAM is ready for accesses. All SRAM banks are enabled/disabled for retention according to values of bits 7:0 of this register 0b = SRAM banks are being set up for retention. Entry into LPM3, LPM4 should not be attempted until this bit is set to 1
15-8	Reserved	R	0h	Reserved. Reads return 0h
7	BNK7_RET <sup>(2)(3)</sup>	RW	1h	0b = Bank7 of the SRAM is not retained in LPM3 or LPM4 1b = Bank7 of the SRAM is retained in LPM3 and LPM4
6	BNK6_RET <sup>(2)(3)</sup>	RW	1h	0b = Bank6 of the SRAM is not retained in LPM3 or LPM4 1b = Bank6 of the SRAM is retained in LPM3 and LPM4
5	BNK5_RET <sup>(2)(3)</sup>	RW	1h	0b = Bank5 of the SRAM is not retained in LPM3 or LPM4 1b = Bank5 of the SRAM is retained in LPM3 and LPM4
4	BNK4_RET <sup>(2)(3)</sup>	RW	1h	0b = Bank4 of the SRAM is not retained in LPM3 or LPM4 1b = Bank4 of the SRAM is retained in LPM3 and LPM4
3	BNK3_RET <sup>(2)(3)</sup>	RW	1h	0b = Bank3 of the SRAM is not retained in LPM3 or LPM4 1b = Bank3 of the SRAM is retained in LPM3 and LPM4
2	BNK2_RET <sup>(2)(3)</sup>	RW	1h	0b = Bank2 of the SRAM is not retained in LPM3 or LPM4 1b = Bank2 of the SRAM is retained in LPM3 and LPM4
1	BNK1_RET <sup>(2)(3)</sup>	RW	1h	0b = Bank1 of the SRAM is not retained in LPM3 or LPM4 1b = Bank1 of the SRAM is retained in LPM3 and LPM4
0	BNK0_RET	R	1h	Bank0 is always retained in LPM3, LPM4 and LPM3.5 modes of operation

<sup>(1)</sup> This bit is automatically set to 0 whenever any of the BNKx\_RET bits in this register are changed. It is set to 1 again after the SRAM controller has recognized the new BNKx\_RET values

<sup>(2)</sup> Value of this bit is a don't care in when the device enters LPM3.5 or LPM4.5 modes of operation. It is always reset, and the SRAM block associated with this bit does not retain its contents.

<sup>(3)</sup> Writes to this bit are allowed ONLY when the SRAM\_RDY bit of this register is set to 1. If the SRAM\_RDY bit is 0, writes to this bit are ignored.

#### **4.11.8 SYS\_FLASH\_SIZE Register (offset = 0020h)**

Flash Size Register

**Figure 4-16. SYS\_FLASH\_SIZE Register**

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

**Table 4-19. SYS\_FLASH\_SIZE Register Description**

Bit	Field	Type	Reset	Description
31-0	SIZE	R	variable	Indicates the size of the Flash main memory on the device. See the device-specific data sheet for the size of flash main memory.

#### 4.11.9 SYS\_DIO\_GLTFLT\_CTL Register (offset = 0030h)

Digital I/O Glitch Filter Control Register

**Figure 4-17. SYS\_DIO\_GLTFLT\_CTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw-1

**Table 4-20. SYS\_DIO\_GLTFLT\_CTL Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved. Always reads 0h
0	GLTCH_EN	RW	1h	0b = Disables glitch filter on the digital I/Os 1b = Enables glitch filter on the digital I/Os

#### **4.11.10 SYS\_SECDATA\_UNLOCK Register (offset = 0040h)**

IP Protected Secure Zone Data Access Unlock Register

**Figure 4-18. SYS\_SECDATA\_UNLOCK**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNLKEY															
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 4-21. SYS\_SECDATA\_UNLOCK Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Reads return 0h
15-0	UNLKEY	RW	0h	Unlock Key, which requests for IP protected secure zone to be unlocked for data accesses. Read back : 0xA596 when unlocked, 0h when locked Write to unlock : 0x695A (only unlocks the same IP protected secure zone as that of the code writing to this register) Other writes : Lock

#### 4.11.11 SYS\_MASTER\_UNLOCK Register (offset = 1000h)

Master Unlock Register

**Figure 4-19. SYS\_MASTER\_UNLOCK Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNLKEY															
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

**Table 4-22. SYS\_MASTER\_UNLOCK Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	NA	Reserved. Reads return 0h
15-0	UNLKEY	RW	0h	Unlock Key, which when written, determines if accesses to SYSCTL registers from offset 1000h are enabled or locked Read back : 0xA596 when unlocked, 0x0 when locked Write to unlock : 0x695A Other writes : Lock

#### **4.11.12 SYS\_BOOTOVER\_REQ0 Register (offset = 1004h)**

Boot Override Request 0 Register

Type = RW. Access allowed only if SYS\_MASTER\_UNLOCK register is unlocked.

**Figure 4-20. SYS\_BOOTOVER\_REQ0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REGVAL															
rw-(0)															
REGVAL															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rw-(0)															

**Table 4-23. SYS\_BOOTOVER\_REQ0 Register Description**

Bit	Field	Type	Reset	Description
31-0	REGVAL	RW	0h	Value set by debugger, read and clear by the CPU

#### 4.11.13 SYS\_BOOTOVER\_REQ1 Register (offset = 1008h)

Boot Override Request 1 Register

Type = RW. Access allowed only if SYS\_MASTER\_UNLOCK register is unlocked.

**Figure 4-21. SYS\_BOOTOVER\_REQ1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REGVAL															
rw-(0)															
REGVAL															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rw-(0)															

**Table 4-24. SYS\_BOOTOVER\_REQ1 Register Description**

Bit	Field	Type	Reset	Description
31-0	REGVAL	RW	0h	Value is set by the debugger, and it is read and cleared by the CPU.

#### **4.11.14 SYS\_BOOTOVER\_ACK Register (offset = 100Ch)**

Boot Override Acknowledge Register

Type = RW. Access allowed only if SYS\_MASTER\_UNLOCK register is unlocked.

**Figure 4-22. SYS\_BOOTOVER\_ACK Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REGVAL															
rw-(0)															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REGVAL															
rw-(0)															

**Table 4-25. SYS\_BOOTOVER\_ACK Register Description**

Bit	Field	Type	Reset	Description
31-0	REGVAL	RW	0h	Value set by CPU, read/clear by the debugger

#### 4.11.15 SYS\_RESET\_REQ Register (offset = 1010h)

Reset Request Register

Type = R/W (with exceptions). Access allowed only if SYS\_MASTER\_UNLOCK register is unlocked.

**Figure 4-23. SYS\_RESET\_REQ Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKEY								Reserved							
w	w	w	w	w	w	w	w	r	r	r	r	r	r	w	w

**Table 4-26. SYS\_RESET\_REQ Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	undefined	Reserved. Reads return 0h
15-8	WKEY	W	undefined	Key to validate/enable write to bits '1-0'. Bits '1-0' are written only if WKEY is 69h in the same write cycle
7-2	Reserved	R	undefined	Reserved. Reads return 0h
1	REBOOT	W	undefined	When written with 1, generates a Reboot Reset pulse to the device Reset Controller
0	POR	W	undefined	When written with 1, generates a POR pulse to the device Reset Controller

#### **4.11.16 SYS\_RESET\_STATOVER Register (offset = 1014h)**

Reset Status and Override Register

Type = R/W. Access allowed only if SYS\_MASTER\_UNLOCK register is unlocked.

**Figure 4-24. SYS\_RESET\_STATOVER Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RBT_OVER	HARD_OVE_R	SOFT_OVE_R	Reserved					REBO OT	HARD	SOFT	
r	r	r	r	r	rw-(0)	rw-(0)	rw-(0)	r	r	r	r	r	r	r	r

**Table 4-27. SYS\_RESET\_STATOVER Register Description**

Bit	Field	Type	Reset	Description
31-11	Reserved	R	undefined	Reserved. Always reads 0h
10	RBT_OVER <sup>(1)(2)</sup>	RW	0h	When 1, activates the override request for the Reboot Reset output of the Reset Controller
9	HARD_OVER <sup>(1)(2)</sup>	RW	0h	When 1, activates the override request for the HARD Reset output of the Reset Controller.
8	SOFT_OVER <sup>(1)(2)</sup>	RW	0h	When 1, activates the override request for the SOFT Reset output of the Reset Controller.
7-3	Reserved	R	0h	Reserved. Always reads 0h
2	REBOOT <sup>(3)</sup>	R	undefined	Indicates if Reboot Reset is asserted
1	HARD <sup>(3)</sup>	R	undefined	Indicates if HARD Reset is asserted
0	SOFT <sup>(3)</sup>	R	undefined	Indicates if SOFT Reset is asserted

<sup>(1)</sup> Overrides are passed to the Reset Controller only if device execution is not secure. If JTAG and SWD lock is active or the CPU is in one of the IP protected secure zones, then reset overrides do not take effect.

<sup>(2)</sup> Overrides are for the corresponding resets only. If a hard reset override is programmed, the soft resets are still propagated into the system. Therefore, the application must program the hard and soft reset bits to override both the resets.

<sup>(3)</sup> The status of resets are still reflected in the system registers even when the reset overrides have been programmed.

#### 4.11.17 SYS\_SYSTEM\_STAT Register (offset = 1020h)

System Status Register

Type = R. Access allowed only if SYS\_MASTER\_UNLOCK register is unlocked.

**Figure 4-25. SYS\_SYSTEM\_STAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										IP_PR OT_A CT	JTAG_ SWD_ LOCK_ ACT	DBG_ SEC_ ACT	Reserv ed	Reserv ed	Reserv ed
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

**Table 4-28. SYS\_SYSTEM\_STAT Register Description**

Bit	Field	Type	Reset	Description
31-6	Reserved	R	undefined	Reserved.
5	IP_PROT_ACT	R	undefined	Indicates if IP protection is active
4	JTAG_SWD_LOCK_A CT	R	undefined	Indicates if JTAG and SWD Lock is active
3	DBG_SEC_ACT	R	undefined	Indicates if the Debug Security is currently active
2-0	Reserved	R	undefined	Reserved.

## **Clock System (CS)**

This chapter describes the operation of the clock system.

Topic	Page
5.1 <b>Clock System Introduction .....</b>	<b>304</b>
5.2 <b>Clock System Operation.....</b>	<b>306</b>
5.3 <b>CS Registers .....</b>	<b>318</b>

## 5.1 Clock System Introduction

The clock system module supports low system cost and low power consumption. The clock module can be configured to operate without any external components, with up to two external crystals, or with resonators, or with an external resistor under full software control.

The clock system module includes the following clock resources:

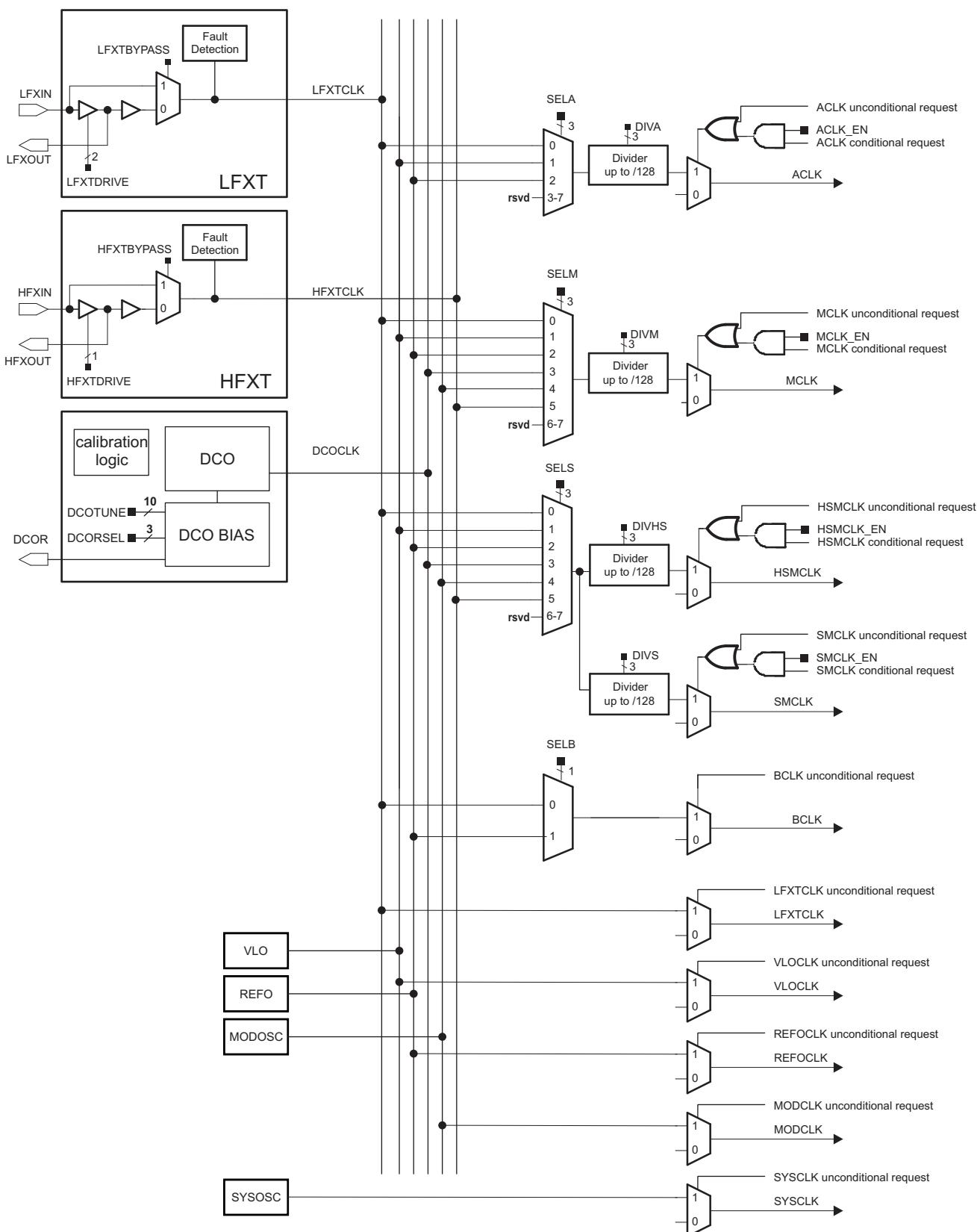
- LFXTCLK: Low-frequency oscillator (LFXT) that can be used either with low-frequency 32768-Hz watch crystals, standard crystals, resonators, or external clock sources in the 32 kHz or below range. When in bypass mode, LFXTCLK can be driven with an external square wave signal in the 32 kHz or below range.
- HFXTCLK: High-frequency oscillator (HFXT) that can be used with standard crystals or resonators in the 1-MHz to 48-MHz range. When in bypass mode, HFXTCLK can be driven with an external square wave signal.
- DCOCLK: Internal digitally controlled oscillator (DCO) with programmable frequencies and 3-MHz frequency by default.
- VLOCLK: Internal very-low-power low-frequency oscillator (VLO) with 9.4-kHz typical frequency
- REFOCLK : Internal, low-power low-frequency oscillator (REFO) with selectable 32.768-kHz or 128-kHz typical frequencies
- MODCLK: Internal low-power oscillator with 25-MHz typical frequency.
- SYSOSC: Internal oscillator with 5MHz typical frequency.

Five primary system clock signals are available from the clock module:

- ACLK: Auxiliary clock. ACLK is software selectable as LFXTCLK, VLOCLK, or REFOCLK. ACLK can be divided by 1, 2, 4, 8, 16, 32, 64 or 128. ACLK is software selectable by individual peripheral modules. ACLK is restricted to maximum frequency of operation of 128 kHz.
- MCLK: Master clock. MCLK is software selectable as LFXTCLK, VLOCLK, REFOCLK, DCOCLK, MODCLK, or HFXTCLK. MCLK can be divided by 1, 2, 4, 8, 16, 32, 64, or 128. MCLK is used by the CPU and peripheral module interfaces, as well as, used directly by some peripheral modules.
- HSMCLK: Subsystem master clock. HSMCLK is software selectable as LFXTCLK, VLOCLK, REFOCLK, DCOCLK, MODCLK, HFXTCLK. HSMCLK can be divided by 1, 2, 4, 8, 16, 32, 64, or 128. HSMCLK is software selectable by individual peripheral modules.
- SMCLK: Low-speed subsystem master clock. SMCLK uses the HSMCLK clock resource selection for its clock resource. SMCLK can be divided independently from HSMCLK by 1, 2, 4, 8, 16, 32, 64, or 128. SMCLK is limited in frequency to half of the rated maximum frequency of HSMCLK. SMCLK is software selectable by individual peripheral modules.
- BCLK: Low-speed backup domain clock. BCLK is software selectable as LFXTCLK and REFOCLK and it is primarily used in the backup domain. BCLK is restricted to a maximum frequency of 32.768 kHz.

VLOCLK, REFOCLK, LFXTCLK, MODCLK, and SYSCLK are additional system clock signals from the clock module. Some of these are not only available as resources to the various system clocks but may also be used directly by various peripheral modules.

Figure 5-1 shows the block diagram of the clock system module.



**Figure 5-1. Clock System Block Diagram**

## 5.2 Clock System Operation

After a system reset, the device enters LDO based active mode at core voltage level 0 (AM\_LDO\_VCORE0). In AM\_LDO\_VCORE0 mode, the CS module default configuration is:

- LFXT crystal operation is selected as the clock resource for LFXTCLK
- LFXTCLK is selected for ACLK (SELAx = 0) and ACLK is undivided (DIVAx = 0)
- LFXTCLK is selected for BCLK (SELB = 0)
- LFXT remains disabled. The crystal pins (LFXIN, LFXOUT) are shared with general-purpose I/Os. LFXIN and LFXOUT pins are set to general-purpose I/Os and LFXT remains disabled until the I/O ports are configured for LFXT operation. To enable LFXT, the PSEL bits associated with the crystal pins must be set. When a 32768-Hz crystal is used for LFXTCLK and a crystal fault is detected, the fault control logic immediately causes ACLK/BCLK to be sourced by REFOCLK (see [Section 5.2.10](#)).
- HFXIN and HFXOUT pins are set to general-purpose I/Os and HFXT is disabled.
- DCOCLK is selected for MCLK, HSMCLK, and SMCLK (SELMx = SELSx = 3) and each system clock is undivided (DIVMx = DIVSx = DIVHSx = 0).

The clock system can be configured or reconfigured by software at any time during program execution. All clock system registers are password protected to prevent inadvertent access except Status Register (CSSTAT) and Interrupt Flag Register (CSIFG).

### 5.2.1 CS Module Features for Low-Power Applications

Conflicting requirements typically exist in low-power applications:

- Low clock frequency for energy conservation and time keeping
- High clock frequency for fast response times and fast burst processing capabilities
- Clock stability over operating temperature and supply voltage
- Low-cost applications with less constrained clock accuracy requirements; for example, crystal-less operation

The CS module addresses these conflicting requirements by allowing the user to select from the five available system clock signals: ACLK, MCLK, HSMCLK, SMCLK, and BCLK. Several clock resources are available to these system clocks. A flexible clock distribution and divider system is provided to fine tune the individual clock requirements.

### 5.2.2 LFXT Oscillator

The LFXT oscillator supports ultra-low-current consumption using a 32768-Hz watch crystal. A watch crystal connects to LFXIN and LFXOUT and requires external capacitors on both terminals. These capacitors should be sized according to the crystal or resonator specifications. Different crystal or resonators are supported by LFXT by choosing the proper LFXTDRIVE settings.

The LFXT pins are shared with general-purpose I/O ports. At power up, the default operation is LFXT crystal operation. However, LFXT remains disabled until the ports shared with LFXT are configured for LFXT operation. The configuration of the shared I/O is determined by the PSEL bit associated with LFXIN and the LFXTBYPASS bit. Setting the PSEL bit causes the LFXIN and LFXOUT ports to be configured for LFXT operation. If LFXTBYPASS is also set, LFXT is configured for bypass mode of operation, and the oscillator associated with LFXT is powered down. In bypass mode of operation, LFXIN can accept an external square-wave clock input signal and LFXOUT is configured as a general-purpose I/O. The PSEL bit associated with LFXOUT is a don't care.

If the PSEL bit associated with LFXIN is cleared, both LFXIN and LFXOUT ports are configured as general-purpose I/Os, and LFXT is disabled.

LFXT is enabled under any of the following conditions:

- For any active mode or LPM0 mode
  - LFXT\_EN = 1.
  - LFXT is a source for ACLK (SELAx = 0).
  - LFXT is a source for BCLK (SELB = 0).

- LFXT is a source for MCLK (SELM<sub>x</sub> = 0).
- LFXT is a source for HSMCLK (SELS<sub>x</sub> = 0).
- LFXT is a source for SMCLK (SELS<sub>x</sub> = 0).
- LFXTCLK is a direct source for any module available in active or LPM0 modes and any LFXTCLK unconditional request is active.
- For LPM3 mode or LPM3.5 mode
  - LFXT\_EN = 1.
  - LFXT is a source for BCLK (SELB = 0) and BCLK request from any backup domain peripheral is active.
  - LFXTCLK is a direct source for any module available in LPM3 or LPM3.5.
- For LPM4.5 mode
  - LFXT is switched off. LFXT\_EN bit has no effect in this mode.

---

**NOTE:** If LFXT is disabled when entering into LPM3 or LPM4 modes, it is not fully enabled and stable upon exit from the selected mode because its enable time is much longer than the wake-up time. If the application requires or desires to keep LFXT enabled during these modes, the LFXT\_EN bit can be set before entering the mode. This causes LFXT to remain enabled. The LFXT\_EN bit has no effect in LPM4.5 mode.

---

### 5.2.3 HFXT Oscillator

The HFXT oscillator can be used with standard crystals or resonators in the 1 MHz to 48 MHz range. HFXT sources HFXTCLK. The HFXTFREQ bits must be set for the appropriate frequency range of operation as shown in [Table 5-1](#) when HFXT operates with external crystals. The HFXTDRIVE bit selects the drive capability of HFXT. The HFXTDRIVE bit must be set to 0 for 1 MHz to 4 MHz operation (HFXTFREQ = 000b) and it must be set to 1 for >4 MHz to 48 MHz operation (HFXTFREQ = 001b to 110b).

**Table 5-1. HFXTFREQ Settings**

HFXT Frequency Range	HFXTFREQ[2:0]
1 to 4 MHz	000
>4 MHz to 8 MHz	001
>8 MHz to 16 MHz	010
>16 MHz to 24 MHz	011
>24 MHz to 32 MHz	100
>32 MHz to 40 MHz	101
>40 MHz to 48 MHz	110

The HFXT pins are shared with general-purpose I/O ports. At power up, the default operation is HFXT crystal operation. However, HFXT remains disabled until the ports shared with HFXT are configured for HFXT operation. The configuration of the shared I/O is determined by the PSEL bit associated with HFXIN and the HFXTBYPASS bit. Setting the PSEL bit causes the HFXIN and HFXOUT ports to be configured for HFXT operation. If HFXTBYPASS is also set, HFXT is configured for bypass mode of operation, and the oscillator associated with HFXT is powered down. In bypass mode of operation, HFXIN can accept an external square-wave clock input signal and HFXOUT is configured as a general-purpose I/O. The PSEL bit associated with HFXOUT is a don't care. Also, HFXTDRIVE and HFXTFREQ bits do not take any effect in the bypass mode of operation.

If the PSEL bit associated with HFXIN is cleared, both HFXIN and HFXOUT ports are configured as general-purpose I/Os, and HFXT is disabled.

HFXT is enabled under any of the following conditions:

- For active modes (AM\_LDO\_VCOREx and AM\_DCDC\_VCOREx) or LPM0 modes (LPM0\_LDO\_VCOREx and LPM0\_DCDC\_VCOREx)

- HFXT\_EN = 1.
- HFXT is a source for MCLK (SELMx = 5).
- HFXT is a source for HSMCLK (SELSx = 5).
- HFXT is a source for SMCLK (SELSx = 5).
- For active modes AM\_LF\_VCOREx or LPM0 modes LPM0\_LF\_VCOREx
  - HFXT is not available and is disabled. HFXT\_EN has no effect.
- For LPM3 or LPM4 or LPM3.5 or LPM4.5 modes
  - HFXT is not available and is disabled. HFXT\_EN has no effect.

### 5.2.3.1 Using HFXT Oscillator After Wakeup From Low-Power Modes

The HFXT oscillator can be used as a clock source before entering LPM3, LPM4, LPM3.5, and LPM4.5 modes. Because the HFXT oscillator is disabled during these low-power modes, the application must reinitialize the HFXT oscillator appropriately after wakeup. The guidelines for enabling the HFXT oscillator after the low-power mode wake-up scenarios are:

**Enabling HFXT oscillator after LPM3 or LPM4 wakeup:** When the device wakes up from LPM3 or LPM4 mode, all clock settings (for example, clock selections and dividers) that were programmed before entry into these low-power modes are retained. The application must restart the HFXT oscillator by clearing the HFXTIFG flag. This is necessary because the HFXT oscillator start-up time is much longer than the wake-up latency from LPM3 and LPM4 modes.

**Enabling HFXT oscillator after LPM3.5 or LPM4.5 wakeup:** When the device wakes up from LPM3.5 or LPM4.5 mode, all clock settings (for example, clock selections and dividers) that were programmed before entry into these low-power modes are lost because of the POR reset. The application must reinitialize the HFXT oscillator completely. This is similar to configuring the HFXT oscillator after device power up. Special care should be taken in this case to ensure that the LOCKLPM5 bit in the PCMCTL1 register is cleared before starting the HFXT oscillator.

In both of the scenarios, the application can suppress interrupt generation by disabling the HFXTIE bit before entering the low-power modes. When the HFXT oscillator is stable, the HFXTIE bit can be reenabled to check for HFXT fault conditions.

### 5.2.4 Internal Very-Low-Power Low-Frequency Oscillator (VLO)

The VLO provides a typical frequency of 9.4-kHz (see device-specific data sheet for parameters) without requiring a crystal. The VLO provides for a low-cost ultra-low-power clock source for applications that do not require an accurate time base. To conserve power, VLO is powered down when not needed and enabled only when required.

VLO is enabled under any of the following conditions:

- For any active mode or LPM0 mode
  - VLO\_EN = 1
  - VLO is a source for ACLK (SELAX = 1).
  - VLO is a source for MCLK (SELMx = 1).
  - VLO is a source for HSMCLK (SELSx = 1).
  - VLO is a source for SMCLK (SELSx = 1).
  - VLOCLK is a direct source for any module available in active mode or LPM0 and any VLOCLK unconditional request active.
- For LPM3 mode or LPM3.5 mode
  - VLO\_EN = 1
  - VLOCLK is a direct source for any module available in LPM3 or LPM3.5.
- For LPM4.5 mode
  - VLO is switched off. VLO\_EN bit has no effect in this mode.

---

**NOTE:** If VLO is disabled when entering into LPM3 or LPM4 modes, it is not fully enabled and stable upon exit from the selected mode because its enable time is much longer than the wake-up time. If the application requires or desires to keep VLO enabled during these modes, the VLO\_EN bit can be set before entering these modes. This causes VLO to remain enabled. The VLO\_EN bit has no effect in LPM4.5 mode.

---

### 5.2.5 Internal Low-Power Low-Frequency Oscillator (REFO)

The REFO provides a typical frequency of 32.768 kHz (see device-specific data sheet for parameters) without requiring a crystal. The REFO provides for a low-cost ultra low-power clock source for applications that do not require a crystal accurate time base, but more accuracy than what can be achieved with the VLO. To conserve power, REFO is powered down when not needed and enabled only when required.

REFO is enabled under any of the following conditions:

- For any active mode or LPM0 mode
  - REFO\_EN = 1
  - REFO is a source for ACLK (SELAx = 2).
  - REFO is a source for BCLK (SELB = 1).
  - REFO is a source for MCLK (SELMx = 2).
  - REFO is a source for HSMCLK (SELSx = 2).
  - REFO is a source for SMCLK (SELSx = 2).
  - REFOCLK is a direct source for any module available in active mode or LPM0 mode and any REFOCLK unconditional request active.
  - LFXTCLK is a source for ACLK (SELAx = 0) and LFXTIFG is set due to crystal oscillator fault.
  - LFXTCLK is a source for BCLK (SELB = 0) and LFXTIFG is set due to crystal oscillator fault.
  - LFXTCLK is a source for MCLK (SELMx = 0) and LFXTIFG is set due to crystal oscillator fault.
  - LFXTCLK is a source for HSMCLK (SELSx = 0) and LFXTIFG is set due to crystal oscillator fault.
  - LFXTCLK is a source for SMCLK (SELSx = 0) and LFXTIFG is set due to crystal oscillator fault.
- For LPM3 mode or LPM3.5 mode
  - REFO\_EN = 1
  - REFO is a source for BCLK (SELB = 1) and BCLK request from any backup domain peripheral is active.
  - REFOCLK is a direct source for any module available in LPM3 or LPM3.5.
- For LPM4.5 mode
  - REFO is switched off. REFO\_EN bit has no effect in this mode.

REFO supports two frequencies of operation: nominally 32.768 kHz (default) and 128 kHz. The frequency is selected by the REFOFSEL bit. Setting REFOFSEL = 1 is useful as a low-power clock source for low-frequency active modes (AM\_LF\_VCOREx) and low-frequency LPM0 modes (LPM0\_LF\_VCOREx). **When REFO is the source for BCLK, it always provides a 32.768-kHz source to BCLK irrespective of the REFOFSEL setting.**

---

**NOTE:** When REFO is turned on directly in 128 kHz mode by setting REFO\_EN = 1 while REFOFSEL = 1 already, the first clock pulse is stretched and its period corresponds to 32.768 kHz clock. The subsequent clock pulses are generated correctly at 128 kHz frequency.

---

**NOTE:** If REFO is disabled when entering into LPM3 or LPM4 modes, it is not fully enabled and stable upon exit from the selected mode because its enable time is much longer than the wake-up time. If the application requires or desires to keep REFO enabled during these modes, the REFO\_EN bit can be set before entering these modes. This causes REFO to remain enabled. The REFO\_EN bit has no effect in LPM4.5 mode.

### 5.2.6 Module Oscillator (MODOSC)

The clock system also supports an internal oscillator, MODOSC, that can be used by MCLK, HSMCLK, SMCLK, and directly by other modules in the system. To conserve power, MODOSC is powered down when not needed and is enabled only when required.

MODOSC is enabled under any of the following conditions:

- For active modes (AM\_LDO\_VCOREx and AM\_DCDC\_VCOREx) or LPM0 modes (LPM0\_LDO\_VCOREx and LPM0\_DCDC\_VCOREx)
  - MODOSC\_EN = 1
  - MODOSC is a source for MCLK (SELMx = 4).
  - MODOSC is a source for HSMCLK (SELsx = 4).
  - MODOSC is a source for SMCLK (SELsx = 4).
  - MODCLK is a direct source for any module available in active mode or LPM0 mode and any MODCLK unconditional request active.
- For LPM3 or LPM4 or LPM3.5 or LPM4.5 mode
  - MODOSC is not available and is disabled. MODOSC\_EN has no effect.

### 5.2.7 System Oscillator (SYSOSC)

Certain modules in the system require an embedded oscillator for general purpose timing, but do not require the stringent accuracy and startup requirements of MODOSC. To conserve power, SYSOSC is powered down when not needed and enabled only when required.

SYSOSC is used by the system for the following purposes:

- Memory controllers (Flash and SRAM) state machine clock.
- Fail-safe clock source for HFXT. (see [Section 5.2.10](#))
- Power control manager (PCM) and power supply system (PSS) state machine clock.
- eUSCI module for clock time-out feature for SMBus support.

### 5.2.8 Digitally Controlled Oscillator (DCO)

The DCO is an integrated digitally controlled oscillator. The DCO has six ranges that are factory calibrated at frequency centers for each respective range. Each range is selectable by the DCORSEL bits. Using the DCOTUNE bits, the application can tune the frequency within a selected range in 0.2% nominal steps to adjust the DCO frequency up or down from the calibrated center frequency. Each range overlaps with its neighboring ranges thereby giving a continuous range of frequencies available for use in the application. The DCO can be used as a source for MCLK, HSMCLK, or SMCLK.

DCO is enabled under any of the following conditions:

- For active modes (AM\_LDO\_VCOREx and AM\_DCDC\_VCOREx) or LPM0 modes (LPM0\_LDO\_VCOREx and LPM0\_DCDC\_VCOREx)
  - DCO\_EN = 1
  - DCO is a source for MCLK (SELMx = 3).
  - DCO is a source for HSMCLK (SELsx = 3).
  - DCO is a source for SMCLK (SELsx = 3).
- For LPM3 or LPM4 or LPM3.5 or LPM4.5 mode
  - DCO is not available and is disabled. DCO\_EN has no effect.

### 5.2.8.1 DCO Modes

The DCO can be operated either in internal resistor or in external resistor modes. The internal resistor mode requires no external components and is the default setting. The external resistor mode requires a resistor to be placed at the DCOR pin to ground. This mode results in higher overall tolerance compared to the internal resistor operation. See the device-specific data sheet for electrical characteristics. The external resistor mode is selected by setting DCORES = 1. Switching to the external resistor mode of operation must always be done at the default DCORSEL setting (DCORSEL = 1). Failure to do so can result in non-deterministic behavior of the device. CSDCOERCAL0, CSDCOERCAL1 registers can be used for calibrating the DCO in the external resistor mode of operation.

### 5.2.8.2 DCO Fault in External Resistor Mode

When the external resistor is detached from DCOR pin thereby forming an open circuit condition either while configuring DCO in external resistor mode (DCORES: 0 to 1) or during run-time when DCO is operating in external resistor mode, a fail-safe mechanism is used to switch DCO from external resistor to internal resistor mode without changing the DCO frequency settings. DCOR\_OPNIFG flag is set in such case and can be used to raise an interrupt. CLR\_DCOR\_OPNIFG bit can be used to clear DCOR\_OPNIFG flag when the DCO is configured back to internal resistor mode (DCORES = 0).

When the DCOR pin gets in contact with the ground thereby forming a short circuit condition either while configuring DCO in external resistor mode (DCORES: 0 to 1) or during run-time when DCO is operating in external resistor mode, a POR reset is triggered into the device. DCOR\_SHTIFG flag is set to indicate the short circuit fault in DCO external resistor configuration. DCOR\_SHT bit in Reset Controller is also set so that the application can determine the cause of POR reset. The corresponding CLR bit Reset Controller can be used to clear DCOR\_SHT bit Reset Controller as well as DCOR\_SHTIFG bit in the Clock System.

### 5.2.8.3 DCO Ranges and Tuning

The DCO has six frequency ranges that can be selected by the DCORSEL bits. Each range overlaps with each of its neighboring ranges to make sure that all frequencies can be selected across the full frequency range. Each frequency range is factory calibrated at the frequency center of the respective range. For example, if a range is selected from 1 MHz to 2 MHz, the default center frequency would be 1.5 MHz, nominal. See [Table 5-4](#) for a list of available range settings. The application can select a different frequency within a range by using the DCOTUNE bits. The DCOTUNE bits are represented in two's complement format and represent the offset from the center frequency. Each LSB represents an offset in the DCO period either positive or negative depending on the sign. By default, the DCOTUNE bits are reset.

Each LSB of DCOTUNE causes a change in the DCO period either positive or negative around this nominal center period, hence decreasing or increasing the overall DCO frequency, respectively. The nominal frequency can be computed using the specified nominal center frequency of interest, along with its factory supplied calibration settings (available in TLV) as follows:

$$F_{DCO,nom} = \frac{F_{RSELx\_CTR,nom}}{1 - \frac{K_{DCOCONST} \times N_{DCOTUNE}}{(1 + K_{DCOCONST} \times (768 - F_{CALCSDCOxRCAL}))}}$$

where

- $F_{DCO,nom}$  = Target Nominal Frequency
- $F_{RSELx\_CTR,nom}$  = Calibrated Nominal Center Frequency for DCO Frequency Range x
- $K_{DCOCONST}$  = DCO Constant (Floating-point value)
- $N_{DCOTUNE}$  = DCO Tune value in decimal
- $F_{CALCSDCOxRCAL}$  = DCO Frequency Calibration value for Range x for Internal or External Resistor modes (3)

Alternatively, DCOTUNE value can be computed from the previous equation by re-arranging it as in [Equation 4](#).

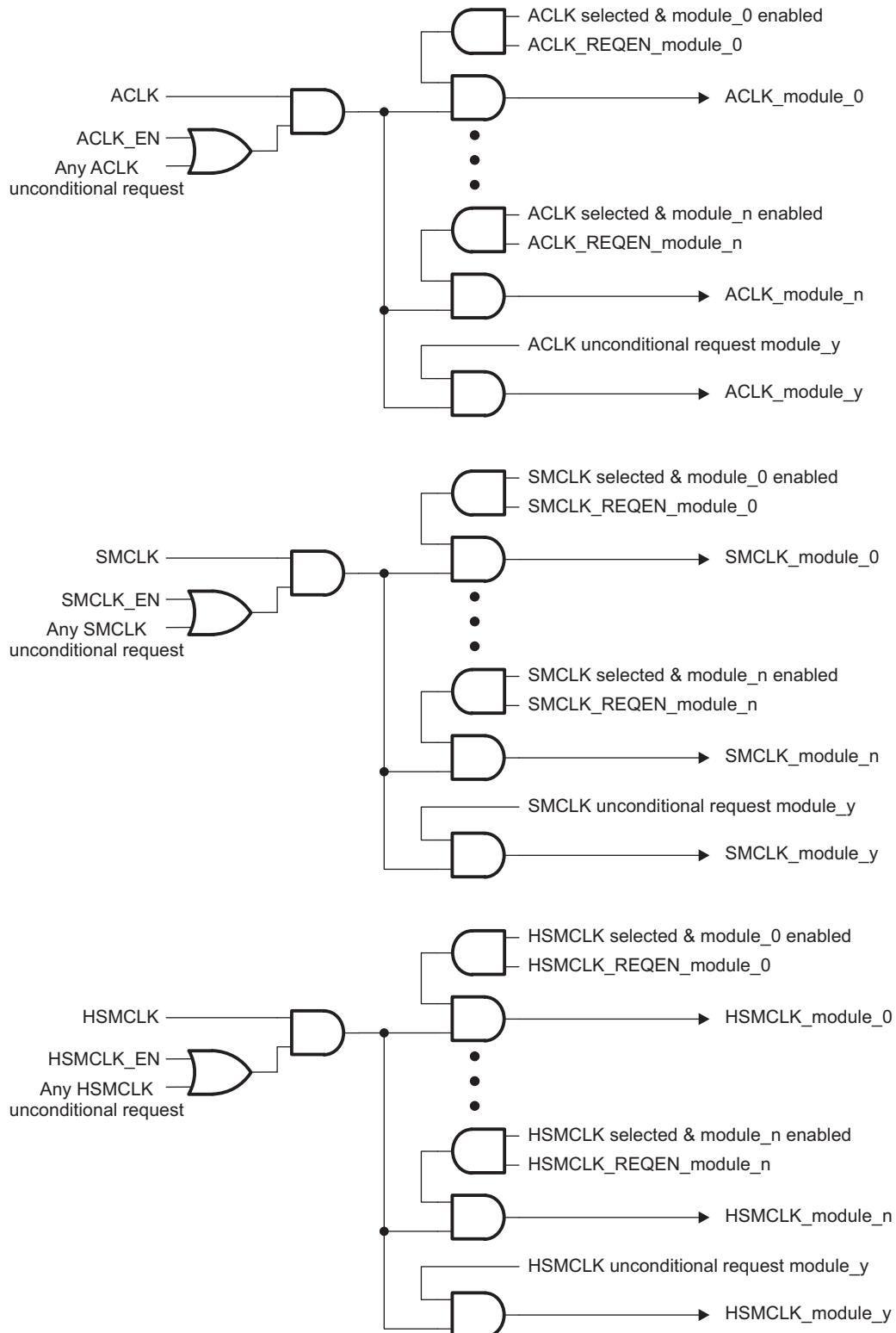
$$N_{DCOTUNE} = \frac{(F_{DCO,nom} - F_{RSELx\_CTR,nom}) \times (1 + K_{DCOCONST} \times (768 - F_{CALCSDCOxRCAL}))}{F_{DCO,nom} \times K_{DCOCONST}} \quad (4)$$

#### 5.2.8.4 Changing DCO Frequency

The DCO frequency can be changed during runtime, however there is a time required for settling of the DCO before the final selected frequency is obtained. There is an inbuilt delay counter in clock system which prevents the clocks to propagate into the system whenever there is a change in system frequency to compensate for the settling time of the DCO. The delay counter kicks in and the DCO clock is stopped for the duration of delay counter for any write access to CSCTL0, CSDCOERCAL0, and CSDCOERCAL1 registers.

#### 5.2.9 Module Clock Request System

A peripheral module requests its clock sources from the clock system if required for its proper operation, regardless of the current power mode of operation, as shown in [Figure 5-2](#).



**Figure 5-2. Module Clock Request System**

A peripheral module may request any system clock (for example, ACLK, HSMCLK, or SMCLK). A request is based on the clock selection and enable of the respective module. For example, if a timer selects ACLK as its clock source and the timer is enabled, the timer generates an ACLK request signal to the clock system. The clock system, in turn, enables ACLK to the module.

The system clocks are each distributed to the peripheral modules as individual clock lines as shown in [Figure 5-2](#). This reduces dynamic power to modules that do not require a particular system clock. Only peripheral clocks that request the respective system clock receive it.

There are two types of clock requests: conditional and unconditional. Conditional requests can be disabled. Unconditional requests cannot be disabled. Each module has an individual clock request enable bit for each clock it can request; for example, ACLKREQ\_EN and HSMCLKREQ\_EN. These bits are used to enable or disable conditional clock requests. Setting these bits enables the conditional clock requests for the module. Clearing these bits causes any conditional clock request to that module to be disabled. Unconditional requests are used by modules that must receive the clock when requested and cannot be disabled; for example, a watchdog timer.

In general, the global clock enable bits (for example, ACLK\_EN) can be used to enable or disable all conditional system clock requests globally. By default these bits are set. Clearing these bits disables the respective system clock even if conditional requests for it are active. If an unconditional request is active, the respective system clock remains active. These bits are useful to disable clocks globally before entering a particular power mode. Note that the respective system clock remains disabled when transitioning back to active mode, and the application must re-enable it if that system clock is desired.

Due to the clock request feature, care must be taken in the application when entering low-power modes to save power. Although the device is programmed to enter the selected low-power mode, a clock request may cause the system to not enter the low-power mode. The software can choose to overlook the active clocks in the system and force a low-power mode entry based on PCM settings. Refer to the PCM specification for details on the forced low-power mode entries.

---

**NOTE:** Care should be taken when enabling or disabling the clock requests to individual modules or globally. Enabling a clock request on an active module immediately causes the requested clock to be presented to the peripheral module. Disabling a clock request on an active module immediately causes any active clock request to be terminated. In both cases, this may cause the status of a module to be affected.

---

### 5.2.10 Clock System Fail-Safe Operation

The oscillator-fault fail-safe feature detects the low frequency and high frequency crystal oscillator faults and DCO external resistor fault as shown in [Figure 5-3](#). The available fault conditions are:

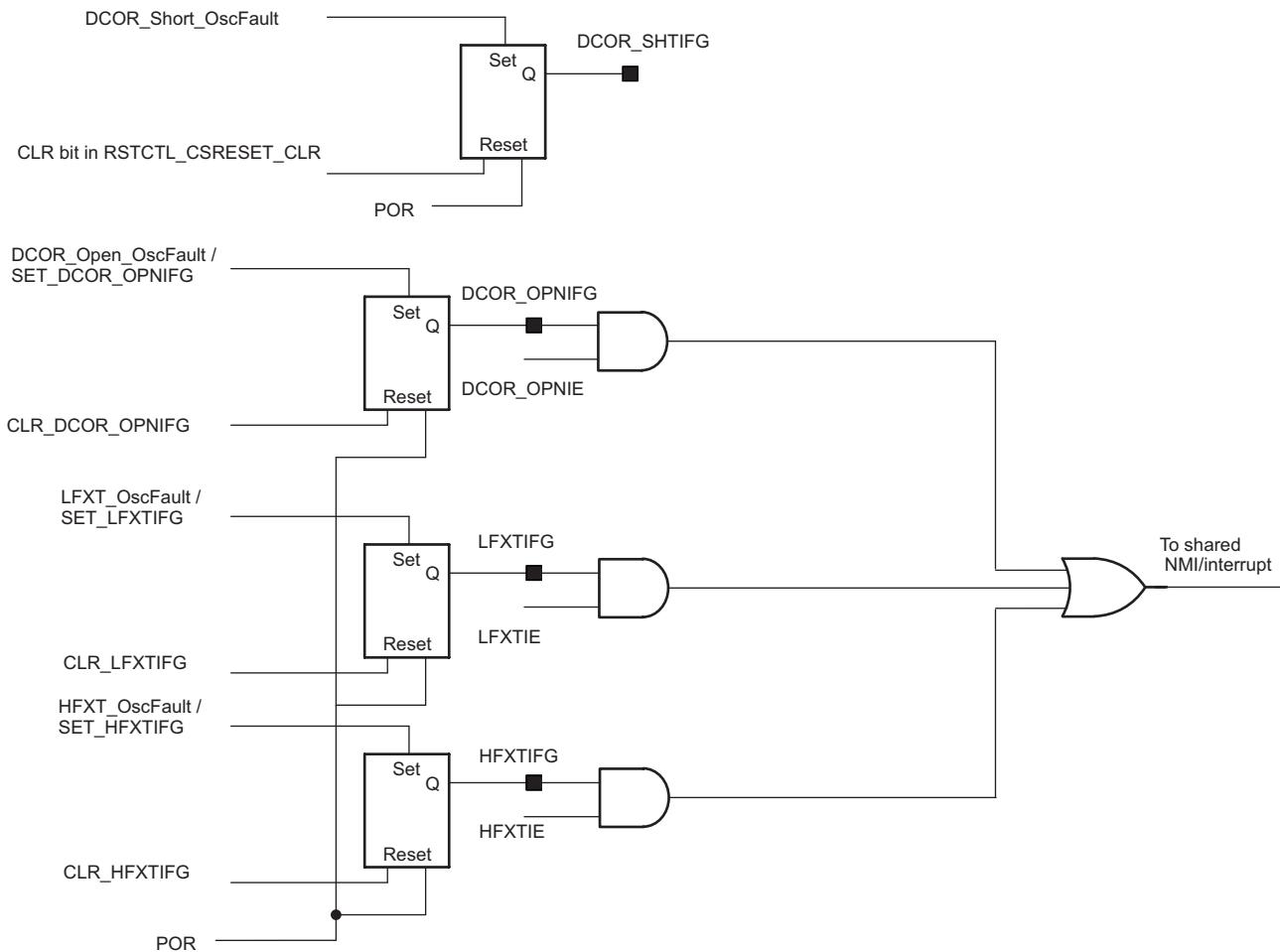
- Low-frequency oscillator fault (LFXTIFG) for LFXT
- High-frequency oscillator fault (HFXTIFG) for HFXT
- DCO external resistor open circuit fault (DCOR\_OPNIFG)
- DCO external resistor short circuit fault (DCOR\_SHTIFG)
- External clock signal faults for all bypass modes

The crystal oscillator fault flags (LFXTIFG, HFXTIFG) are set if the corresponding crystal oscillator is turned on but is not operating properly. After a fault flag is set, it remains set until reset by software, even if the fault condition no longer exists. If the application clears the corresponding fault flag(s), and the fault condition(s) still exists, the fault flag(s) are automatically set again, otherwise they remain cleared.

The oscillator-fault interrupt flags are set and latched at POR or when any oscillator fault is detected. When any of the fault flags are set and the corresponding interrupt enables are set, an NMI/interrupt request is initiated by the CS to the SYSTCL module. The fault flags must be cleared by software. The source of the fault can be identified by checking the individual fault flags.

If LFXT is sourcing any system clock (ACLK, BCLK, MCLK, HSMCLK, SMCLK, or LFXTCLK) and a fault is detected, the system clock is automatically switched to REFO for its clock source. **The REFO clock in fail-safe mode of operation always runs at 32.768-kHz and the REFOFSEL bit setting does not take any effect.** The LFXT fault logic works in all power modes, including LPM3 mode. Similarly, If HFXT is sourcing MCLK, HSMCLK, or SMCLK, and a fault is detected, the system clocks are automatically switched to SYSOSC for their clock source. The HFXT fault logic does not work in low-power modes as the high-frequency operation in low-power modes is not supported.

The fail-safe logic does not change the SELA, SELB, SELM, and SELS bit settings. The fail-safe mechanism behaves the same in normal and bypass modes.



**Figure 5-3. Oscillator Fault Logic**

**NOTE: Fault conditions**

**LFXT\_OscFault:** This signal is set after the LFXT oscillator has stopped operation and is cleared after operation resumes. The fault condition causes LFXTIFG to be set and remain set. If the user clears LFXTIFG and the fault condition still exists, LFXTIFG is set again.

**HFXT\_OscFault:** This signal is set after the HFXT oscillator has stopped operation and is cleared after operation resumes. The fault condition causes HFXTIFG to be set and remain set. If the user clears HFXTIFG and the fault condition still exists, HFXTIFG is set again.

**DCOR\_Open\_OscFault:** This signal is set upon DCO external resistor open circuit fault. The fault condition causes DCOR\_OPNIFG to be set and remain set. If the user clears DCOR\_OPNIFG and the fault condition still exists, DCOR\_OPNIFG is set again.

**DCOR\_Short\_OscFault:** This signal is set upon DCO external resistor short circuit fault. The fault condition causes DCOR\_SHTIFG to be set and leads to device POR. DCO\_SHTIFG is cleared upon device power cycle or by writing into CLR bit of RSTCTL\_CSRESET\_CLR register.

**NOTE: Fault logic**

As long as a fault condition still exists, the individual fault flags remains set. The clock logic switches back to the original user settings before the fault condition, once the fault condition is resolved and the corresponding fault flag is cleared.

### 5.2.11 Start-Up Counters

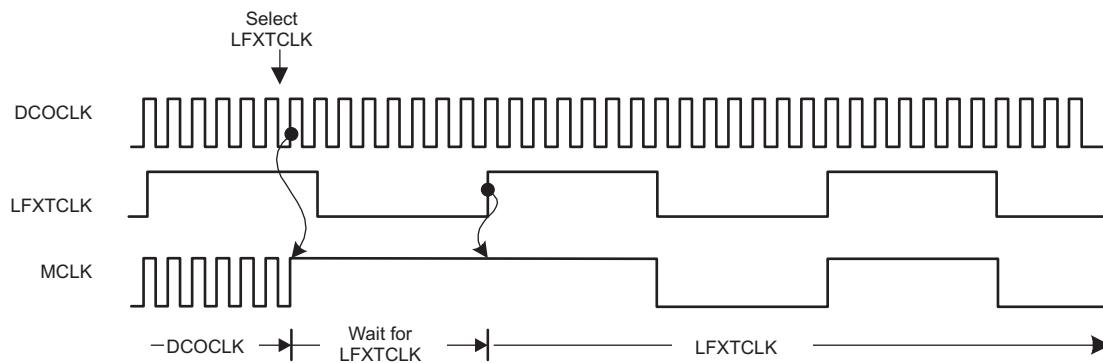
The LFXT includes a counter that makes sure that a programmable number of clock cycles have passed before the LFXT\_OscFault signal is cleared. The counter can be programmed from 4096 to 32768 counts using the FCNTLF bits. The default is the maximum count. Any crystal fault restarts the counter. It is also possible to restart the counter directly via software by setting RCNTLF. RCNTLF bit is self-clearing. When it is written, the counter immediately restarts its count. For applications that do not require the startup counter, it can be disabled by clearing FCNTLF\_EN. Clearing FCNTLF\_EN immediately clears and halts the timer. Disabling the counters does not disable the fault logic. The counter is available in both normal and bypass modes of operation.

Similarly, HFXT includes a counter that makes sure that a programmable number of clock cycles have passed before the HFXT\_OscFault signal is cleared. The counter can be programmed from 2048 to 16384 counts using the FCNTHF bits. The default is the maximum count. Any crystal fault restarts the counter. It is also possible to restart the counter directly via software by setting RCNTHF. RCNTHF bit is self-clearing. Once written, the counter immediately restarts its count. For applications that do not require the startup counter, it can be disabled by clearing FCNTHF\_EN. Clearing FCNTHF\_EN immediately clears and halts the timer. Disabling the counters does not disable the fault logic. The counter is available in both normal and bypass modes of operation.

### 5.2.12 Synchronization of Clock Signals

When switching ACLK, MCLK, HSMCLK, or SMCLK from one clock source to the another, the switch is synchronized to avoid critical race conditions as shown in [Figure 5-4](#):

- The current clock cycle continues until the next rising edge.
- The clock remains high until the next rising edge of the new clock.
- The new clock source is selected and continues with a full high period.



**Figure 5-4. Switch MCLK from DCOCLK to LFXTCLK**

### 5.2.13 Clock Status

The CSSTAT is a read-only register that can be used by the application to determine what system clocks and system resources are currently active. It is possible that a system resource could be active, yet a corresponding system clock that has selected the resource is not. For example, the clock resource may be REFO and ACLK has selected REFO as its source. In addition, the user has set REFO\_EN = 1. If there are no modules actively using ACLK, REFO is shown as active (REFO\_ON = 1), but ACLK is shown as inactive (ACKL\_ON = 0).

In addition, there are READY bits corresponding to each clock tree. Before the change in frequency or the source oscillator for a particular clock tree, the application must ensure that the current clock settings are stable by polling on the READY bits. For example, if the application intends to switch the ACLK source from REFOCLK to LFXTCLK, it must ensure that the current setting for ACLK (out of REFO) is ready by polling the ACLK\_READY in the CSSTAT register and then initiate the SELA change to LFXT. Then ACLK\_READY bit can be polled again to ascertain that ACLK is sourced out of LFXT. This is to ensure that the clock system does not go into a nondeterministic state. If the application changes the clock settings for ACLK from REFO to LFXT (assume ACLK\_READY = 1) and then immediately changes the SELA to use VLO while the REFO to LFXT change was ongoing (ACKL\_READY = 0), then the clock system can get into a nondeterministic state.

---

**NOTE:** Before entering into any of the low-power modes, the user software must ensure that the system clocks have been set to the correct frequency of operation. The application must ascertain this by checking the clock ready bits in the CSSTAT register. Failure to check the clocks being ready before entering into low-power modes, may result in the system being in a nondeterministic state.

After a change in clock source or clock tree or clock frequency/divider settings, **10 CPU bus clock cycles** are required until the correct status is captured in the CSSTAT register. Application must account for this latency while checking the clock configuration status in the CSSTAT register.

---

**NOTE:** If any outstanding power-mode transition is ongoing, then the PCM locks the clock system registers until the power-mode transition is complete. This is indicated in the PCM module through the PMR\_BUSY bit in the PCMCTL1 register.

---

### 5.3 CS Registers

[Table 5-2](#) shows the CS module registers and their address offsets. The base address can be found in the device-specific data sheet.

All CS registers are password protected except Status Register (CSSTAT) and Interrupt Flag Register (CSIFG). The password that is defined in CSKEY controls access to the CS registers. To enable write access to the CS registers, write the correct password to CSKEY. To disable write access, write any incorrect password to CSKEY.

**Table 5-2. CS Registers**

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	CSKEY	Key Register	Read/write	Word	0000_A596h	<a href="#">Section 5.3.1</a>
04h	CSCTL0	Control 0 Register	Read/write	Word	0001_0000h	<a href="#">Section 5.3.2</a>
08h	CSCTL1	Control 1 Register	Read/write	Word	0000_0033h	<a href="#">Section 5.3.3</a>
0Ch	CSCTL2	Control 2 Register	Read/write	Word	0001_0003h	<a href="#">Section 5.3.4</a>
10h	CSCTL3	Control 3 Register	Read/write	Word	0000_00BBh	<a href="#">Section 5.3.5</a>
30h	CSCLKEN	Clock Enable Register	Read/write	Word	0000_000Fh	<a href="#">Section 5.3.6</a>
34h	CSSTAT	Status Register	Read	Word	0000_0003h	<a href="#">Section 5.3.7</a>
40h	CSIE	Interrupt Enable Register	Read/write	Word	0000_0000h	<a href="#">Section 5.3.8</a>
48h	CSIFG	Interrupt Flag Register	Read	Word	0000_0001h	<a href="#">Section 5.3.9</a>
50h	CSCLRIFG	Clear Interrupt Flag Register	Write	Word	0000_0000h	<a href="#">Section 5.3.10</a>
58h	CSSETIFG	Set Interrupt Flag Register	Write	Word	0000_0000h	<a href="#">Section 5.3.11</a>
60h	CSDCOERCAL0	DCO External Resistor Calibration 0 Register	Read/Write	Word	0100_0000h	<a href="#">Section 5.3.12</a>
64h	CSDCOERCAL1	DCO External Resistor Calibration 1 Register	Read/Write	Word	0000_0100h	<a href="#">Section 5.3.13</a>

---

**NOTE:** This is a 32-bit module and can be accessed through word (32-bit) or half-word (16-bit) or byte (8-bit) accesses.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### **5.3.1 CSKEY Register (offset = 00h) [reset = 0000\_A596h]**

Clock System Key Register

**Figure 5-5. CSKEY Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
CSKEY							
rw-1	rw-0	rw-1	rw-0	rw-0	rw-1	rw-0	rw-1
7	6	5	4	3	2	1	0
CSKEY							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0

**Table 5-3. CSKEY Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads as 0.
15- 0	CSKEY	RW	A596h	Write CSKEY = xxxx_695Ah to unlock CS registers. All 16 LSBs need to be written together. Writing CSKEY with any other value causes CS registers to be locked and any writes to these registers are ignored, while reads are still performed. Always reads back A596h.

### 5.3.2 CSCTL0 Register (offset = 04h) [reset = 0001\_0000h]

Clock System Control 0 Register

**Figure 5-6. CSCTL0 Register**

31	30	29	28	27	26	25	24
Reserved							Reserved
r-0	r-0	r-0	r-0	r-0	r-0	r-0	rw-0
23	22	21	20	19	18	17	16
DCOEN	DCORES	Reserved			DCORSEL		
rw-0	rw-0	r-0	r-0	r-0	rw-0	rw-0	rw-1
15	14	13	12	11	10	9	8
Reserved		Reserved			DCOTUNE		
r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
DCOTUNE							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 5-4. CSCTL0 Register Description**

Bit	Field	Type	Reset	Description
31-25	Reserved	R	0h	Reserved. Always reads as 0.
24	Reserved	RW	0h	Reserved. Must be written as zero.
23	DCOEN	RW	0h	Enables the DCO oscillator regardless if used as a clock resource. 0b = DCO is on if it is used as a source for MCLK, HSMCLK , or SMCLK and clock is requested , otherwise it is disabled. 1b = DCO is on.
22	DCORES	RW	0h	Enables the DCO external resistor mode. 0b = Internal resistor mode 1b = External resistor mode
21-19	Reserved	R	0h	Reserved. Always reads as 0.
18-16	DCORSEL	RW	1h	DCO frequency range select. Selects frequency range settings for the DCO. 000b = Nominal DCO Frequency (MHz): 1.5; Nominal DCO Frequency Range (MHz): 1 to 2 001b = Nominal DCO Frequency (MHz): 3; Nominal DCO Frequency Range (MHz): 2 to 4 010b = Nominal DCO Frequency (MHz): 6; Nominal DCO Frequency Range (MHz): 4 to 8 011b = Nominal DCO Frequency (MHz): 12; Nominal DCO Frequency Range (MHz): 8 to 16 100b = Nominal DCO Frequency (MHz): 24; Nominal DCO Frequency Range (MHz): 16 to 32 101b = Nominal DCO Frequency (MHz): 48; Nominal DCO Frequency Range (MHz): 32 to 64 110b to 111b = Nominal DCO Frequency (MHz): Reserved--defaults to 1.5 when selected; Nominal DCO Frequency Range (MHz): Reserved--defaults to 1 to 2 when selected.
15-13	Reserved	R	0h	Reserved. Always reads as 0.
12-10	Reserved	RW	0h	Reserved. Must be written as zero.
9-0	DCOTUNE	RW	0h	DCO frequency tuning select. 2s complement representation. Value represents an offset from the calibrated center frequency for the range selected by the DCORSEL bits.

### 5.3.3 CSCTL1 Register (offset = 08h) [reset = 0000\_0033h]

Clock System Control 1 Register

**Figure 5-7. CSCTL1 Register**

31	30	29	28	27	26	25	24
Reserved		DIVS		Reserved		DIVA	
r-0	rw-0	rw-0	rw-0	r-0	rw-0	rw-0	rw-0
23	22	21	20	19	18	17	16
Reserved		DIVHS		Reserved		DIVM	
r-0	rw-0	rw-0	rw-0	r-0	rw-0	rw-0	rw-0
15	14	13	12	11	10	9	8
Reserved		Reserved	SELB	Reserved		SELA	
r-0	r-0	r-0	rw-0	r-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved		SELS		Reserved		SELM	
r-0	rw-0	rw-1	rw-1	r-0	rw-0	rw-1	rw-1

**Table 5-5. CSCTL1 Register Description**

Bit	Field	Type	Reset	Description
31	Reserved	R	0h	Reserved. Always reads as 0.
30-28	DIVS	RW	0h	SMCLK source divider. 000b = f(SMCLK)/1 001b = f(SMCLK)/2 010b = f(SMCLK)/4 011b = f(SMCLK)/8 100b = f(SMCLK)/16 101b = f(SMCLK)/32 110b = f(SMCLK)/64 111b = f(SMCLK)/128
27	Reserved	R	0h	Reserved. Always reads as 0.
26-24	DIVA	RW	0h	ACLK source divider. 000b = f(ACLK)/1 001b = f(ACLK)/2 010b = f(ACLK)/4 011b = f(ACLK)/8 100b = f(ACLK)/16 101b = f(ACLK)/32 110b = f(ACLK)/64 111b = f(ACLK)/128
23	Reserved	R	0h	Reserved. Always reads as 0.
22-20	DIVHS	RW	0h	HSMCLK source divider. 000b = f(HSMCLK)/1 001b = f(HSMCLK)/2 010b = f(HSMCLK)/4 011b = f(HSMCLK)/8 100b = f(HSMCLK)/16 101b = f(HSMCLK)/32 110b = f(HSMCLK)/64 111b = f(HSMCLK)/128
19	Reserved	R	0h	Reserved. Always reads as 0.

**Table 5-5. CSCTL1 Register Description (continued)**

Bit	Field	Type	Reset	Description
18-16	DIVM	RW	0h	MCLK source divider. 000b = f(MCLK)/1 001b = f(MCLK)/2 010b = f(MCLK)/4 011b = f(MCLK)/8 100b = f(MCLK)/16 101b = f(MCLK)/32 110b = f(MCLK)/64 111b = f(MCLK)/128
15-13	Reserved	R	0h	Reserved. Always reads as 0.
12	SELB	RW	0h	Selects the BCLK source. 0b = LFXTCLK 1b = REFOCLK
11	Reserved	R	0h	Reserved. Always reads as 0.
10-8	SELA	RW	0h	Selects the ACLK source. 000b = LFXTCLK 001b = VLOCLK 010b = REFOCLK 011b-111b = Reserved for future use. Defaults to REFOCLK. Not recommended for use to ensure future compatibilities.
7	Reserved	R	0h	Reserved. Always reads as 0.
6-4	SELS	RW	3h	Selects the SMCLK and HSMCLK source. 000b = LFXTCLK 001b = VLOCLK 010b = REFOCLK 011b = DCOCLK 100b = MODOSC 101b = HFXTCLK 110b-111b = Reserved for future use. Defaults to DCOCLK. Not recommended for use to ensure future compatibilities.
3	Reserved	R	0h	Reserved. Always reads as 0.
2-0	SELM	RW	3h	Selects the MCLK source. 000b = LFXTCLK 001b = VLOCLK 010b = REFOCLK 011b = DCOCLK 100b = MODOSC 101b = HFXTCLK 110b-111b = Reserved for future use. Defaults to DCOCLK. Not recommended for use to ensure future compatibilities.

### 5.3.4 CSCTL2 Register (offset = 0Ch) [reset = 0001\_0003h]

Clock System Control 2 Register

**Figure 5-8. CSCTL2 Register**

31	30	29	28	27	26	25	24
Reserved					HFXTBYPASS	HFXT_EN	
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0
23	22	21	20	19	18	17	16
Reserved	HFXTFREQ			Reserved	Reserved	HFXTDRIVE	
r-0	rw-0	rw-0	rw-0	r-0	rw-0	rw-0	rw-1
15	14	13	12	11	10	9	8
Reserved					LFXTBYPASS	LFXT_EN	
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved	Reserved			LFXTDRIVE			
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1	rw-1

**Table 5-6. CSCTL2 Register Description**

Bit	Field	Type	Reset	Description
31-26	Reserved	R	0h	Reserved. Always reads as 0.
25	HFXTBYPASS	RW	0h	HFXT bypass select. 0b = HFXT sourced by external crystal. 1b = HFXT sourced by external square wave.
24	HFXT_EN	RW	0h	Turns on the HFXT oscillator regardless if used as a clock resource. 0b = HFXT is on if it is used as a source for MCLK, HSMCLK , or SMCLK and is selected via the port selection and not in bypass mode of operation. 1b = HFXT is on if HFXT is selected via the port selection and HFXT is not in bypass mode of operation.
23	Reserved	R	0h	Reserved. Always reads as 0.
22-20	HFXTFREQ	RW	0h	HFXT frequency selection. These bits must be set to the appropriate value based on the frequency of the crystal connected. These bits are don't care in the HFXT bypass mode of operation. 000b = 1 MHz to 4 MHz 001b = >4 MHz to 8 MHz 010b = >8 MHz to 16 MHz 011b = >16 MHz to 24 MHz 100b = >24 MHz to 32 MHz 101b = >32 MHz to 40 MHz 110b = >40 MHz to 48 MHz 111b = Reserved for future use.
19	Reserved	R	0h	Reserved. Always reads as 0.
18-17	Reserved	RW	0h	Reserved.
16	HFXTDRIVE	RW	1h	HFXT oscillator drive selection. Reset value is 1h when HFXT available, and 0h when HFXT not available. This bit is a don't care in the HFXT bypass mode of operation. 0b = To be used for HFXTFREQ setting 000b 1b = To be used for HFXTFREQ settings 001b to 110b
15-10	Reserved	R	0h	Reserved. Always reads as 0.
9	LFXTBYPASS	RW	0h	LFXT bypass select. 0b = LFXT sourced by external crystal. 1b = LFXT sourced by external square wave.

**Table 5-6. CSCTL2 Register Description (continued)**

Bit	Field	Type	Reset	Description
8	LFXT_EN	RW	0h	Turns on the LFXT oscillator regardless if used as a clock resource. 0b = LFXT is on if it is used as a source for ACLK, MCLK, HSMCLK , or SMCLK and is selected via the port selection and not in bypass mode of operation. 1b = LFXT is on if LFXT is selected via the port selection and LFXT is not in bypass mode of operation.
7	Reserved	RW	0h	Reserved. Must be written as zero.
6-3	Reserved	RW	0h	Reserved. Must be written as zero.
2	Reserved	RW	0h	Reserved.
1-0	LFXTDRIVE	RW	3h	The LFXT oscillator current can be adjusted to its drive needs. 0h = Lowest drive strength and current consumption LFXT oscillator. 1h = Increased drive strength LFXT oscillator. 2h = Increased drive strength LFXT oscillator. 3h = Maximum drive strength and maximum current consumption LFXT oscillator.

### 5.3.5 CSCTL3 Register (offset = 10h) [reset = 0000\_00BBh]

Clock System Control 3 Register

**Figure 5-9. CSCTL3 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
FCNTHF_EN	RFCNTHF	FCNTHF		FCNTLF_EN	RFCNTLF	FCNTLF	
rw-(1)	w-0	rw-(1)		rw-(1)	w-0	rw-(1)	

**Table 5-7. CSCTL3 Register Description**

Bit	Field	Type	Reset	Description
31-8	Reserved	R	0h	Reserved. Always reads as 0.
7	FCNTHF_EN	RW	1h	Enable start fault counter for HFXT. 0b = Startup fault counter disabled. Counter is cleared. 1b = Startup fault counter enabled
6	RFCNTHF	W	0h	Reset start fault counter for HFXT. Write 1 only. Self clears once written. 0b = Not applicable. Always reads as zero due to self clearing. 1b = Restarts the counter immediately.
5-4	FCNTHF	RW	3h	Start flag counter for HFXT. Selects number of HFXT cycles before HFXTIFG can be cleared. 00b = 2048 cycles 01b = 4096 cycles 10b = 8192 cycles 11b = 16384 cycles
3	FCNTLF_EN	RW	1h	Enable start fault counter for LFXT. 0b = Startup fault counter disabled. Counter is cleared. 1b = Startup fault counter enabled
2	RFCNTLF	W	0h	Reset start fault counter for LFXT. Write 1 only. Self clears once written. 0b = Not applicable. Always reads as zero due to self clearing. 1b = Restarts the counter immediately.
0-1	FCNTLF	RW	3h	Start flag counter for LFXT. Selects number of LFXT cycles before LFXTIFG can be cleared. 00b = 4096 cycles 01b = 8192 cycles 10b = 16384 cycles 11b = 32768 cycles

### 5.3.6 CSCLKEN Register (offset = 30h) [reset = 0000\_000Fh]

Clock System Clock Enable Register

**Figure 5-10. CSCLKEN Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
REFOFSEL	Reserved				MODOSC_EN	REFO_EN	VLO_EN
rw-0	r-0	r-0	r-0	r-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved	Reserved			SMCLK_EN	HSMCLK_EN	MCLK_EN	ACLK_EN
r-0	r-0	r-0	r-0	rw-1	rw-1	rw-1	rw-1

**Table 5-8. CSCLKEN Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads as 0.
15	REFOFSEL	RW	0h	Selects REFO nominal frequency. 0b = 32.768 kHz 1b = 128 kHz
14-11	Reserved	R	0h	Reserved. Always reads as 0.
10	MODOSC_EN	RW	0h	Turns on the MODOSC oscillator regardless if used as a clock resource. 0b = MODOSC is on only if it is used as a source for ACLK, MCLK, HSMCLK , or SMCLK. 1b = MODOSC is on
9	REFO_EN	RW	0h	Turns on the REFO oscillator regardless if used as a clock resource. 0b = REFO is on only if it is used as a source for ACLK, MCLK, HSMCLK , or SMCLK. 1b = REFO is on
8	VLO_EN	RW	0h	Turns on the VLO oscillator regardless if used as a clock resource. 0b = VLO is on only if it is used as a source for ACLK, MCLK, HSMCLK , or SMCLK. 1b = VLO is on
7-4	Reserved	R	0h	Reserved. Always reads as 0.
3	SMCLK_EN	RW	1h	SMCLK system clock conditional request enable 0b = SMCLK disabled regardless of conditional clock requests 1b = SMCLK enabled based on any conditional clock requests.
2	HSMCLK_EN	RW	1h	HSMCLK system clock conditional request enable 0b = HSMCLK disabled regardless of conditional clock requests 1b = HSMCLK enabled based on any conditional clock requests.
1	MCLK_EN	RW	1h	MCLK system clock conditional request enable 0b = MCLK disabled regardless of conditional clock requests 1b = MCLK enabled based on any conditional clock requests.
0	ACLK_EN	RW	1h	ACLK system clock conditional request enable 0b = ACLK disabled regardless of conditional clock requests 1b = ACLK enabled based on any conditional clock requests.

### 5.3.7 CSSTAT Register (offset = 34h) [reset = 0000\_0003h]

Clock System Status Register

**Figure 5-11. CSSTAT Register**

31	30	29	28	27	26	25	24
		Reserved	BCLK_READY	SMCLK_READY	HSMCLK_READY	MCLK_READY	ACLK_READY
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
23	22	21	20	19	18	17	16
REFOCLK_ON	LFXTCLK_ON	VLOCLK_ON	MODCLK_ON	SMCLK_ON	HSMCLK_ON	MCLK_ON	ACLK_ON
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
			Reserved				
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
REFO_ON	LFXT_ON	VLO_ON	MODOSC_ON	Reserved	HFXT_ON	DCOBIAST_ON	DCO_ON
r-0	r-0	r-0	r-0	r-0	r-0	r-1	r-1

**Table 5-9. CSSTAT Register Description**

Bit	Field	Type	Reset	Description
31-29	Reserved	R	0h	Reserved. Always reads as 0.
28	BCLK_READY	R	0h	BCLK Ready status. This bit indicates whether the clock is stable after a change in the frequency settings. 0b = Not ready 1b = Ready
27	SMCLK_READY	R	0h	SMCLK Ready status. This bit indicates whether the clock is stable after a change in the frequency/divider settings. 0b = Not ready 1b = Ready
26	HSMCLK_READY	R	0h	HSMCLK Ready status. This bit indicates whether the clock is stable after a change in the frequency/divider settings. 0b = Not ready 1b = Ready
25	MCLK_READY	R	0h	MCLK Ready status. This bit indicates whether the clock is stable after a change in the frequency/divider settings. 0b = Not ready 1b = Ready
24	ACLK_READY	R	0h	ACLK Ready status. This bit indicates whether the clock is stable after a change in the frequency/divider settings. 0b = Not ready 1b = Ready
23	REFOCLK_ON	R	0h	REFOCLK system clock status 0b = Inactive 1b = Active
22	LFXTCLK_ON	R	0h	LFXTCLK system clock status 0b = Inactive 1b = Active
21	VLOCLK_ON	R	0h	VLOCLK system clock status 0b = Inactive 1b = Active
20	MODCLK_ON	R	0h	MODCLK system clock status 0b = Inactive 1b = Active

**Table 5-9. CSSTAT Register Description (continued)**

Bit	Field	Type	Reset	Description
19	SMCLK_ON	R	0h	SMCLK system clock status 0b = Inactive 1b = Active
18	HSMCLK_ON	R	0h	HSMCLK system clock status 0b = Inactive 1b = Active
17	MCLK_ON	R	0h	MCLK system clock status 0b = Inactive 1b = Active
16	ACLK_ON	R	0h	ACLK system clock status 0b = Inactive 1b = Active
15-8	Reserved	R	0h	Reserved. Always reads as 0.
7	REFO_ON	R	0h	REFO status 0b = Inactive 1b = Active
6	LFXT_ON	R	0h	LFXT status. 0b = Inactive 1b = Active
5	VLO_ON	R	0h	VLO status 0b = Inactive 1b = Active
4	MODOSC_ON	R	0h	MODOSC status 0b = Inactive 1b = Active
3	Reserved	R	0h	Reserved. Always reads as 0.
2	HFXT_ON	R	0h	HFXT status. 0b = Inactive 1b = Active
1	DCOBIAST_ON	R	1h	DCO bias status 0b = Inactive 1b = Active
0	DCO_ON	R	1h	DCO status 0b = Inactive 1b = Active

### 5.3.8 CSIE Register (offset = 40h) [reset = 0000\_0000h]

Clock System Interrupt Enable Register

**Figure 5-12. CSIE Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	FCNTHFIE	FCNTLFIE
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0
7	6	5	4	3	2	1	0
Reserved	DCOR_OPNIE	Reserved	Reserved	Reserved	Reserved	HFXTIE	LFXTIE
r-0	rw-0	rw-0	rw-0	r-0	r-0	rw-0	rw-0

**Table 5-10. CSIE Register Description**

Bit	Field	Type	Reset	Description
31-10	Reserved	R	0h	Reserved. Always reads as 0.
9	FCNTHFIE	RW	0h	Start fault counter interrupt enable HFXT. 0b = Interrupt disabled 1b = Interrupt enabled
8	FCNTLFIE	RW	0h	Start fault counter interrupt enable LFXT. 0b = Interrupt disabled 1b = Interrupt enabled
7	Reserved	R	0h	Reserved. Always reads as 0.
6	DCOR_OPNIE	RW	0h	DCO external resistor open circuit fault flag interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
5	Reserved	RW	0h	Reserved.
4	Reserved	RW	0h	Reserved.
3	Reserved	R	0h	Reserved. Always reads as 0.
2	Reserved	R	0h	Reserved. Always reads as 0.
1	HFXTIE	RW	0h	HFXT oscillator fault flag interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
0	LFXTIE	RW	0h	LFXT oscillator fault flag interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled

### 5.3.9 CSIFG Register (offset = 48h) [reset = 0000\_0001h]

Clock System Interrupt Flag Register

**Figure 5-13. CSIFG Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	FCNTHIFG	FCNTLFIFG
r-0	r-0	r-0	r-0	r-0	r-0	r-(0)	r-(0)
7	6	5	4	3	2	1	0
Reserved	DCOR_OPNIF <sub>G</sub>	DCOR_SHTIF <sub>G</sub>	Reserved	Reserved	Reserved	HFXTIFG	LFXTIFG
r-0	r-(0)	r-(0)	r-(0)	r-0	r-0	r-(0)	r-(1)

**Table 5-11. CSIFG Register Description**

Bit	Field	Type	Reset	Description
31-10	Reserved	R	0h	Reserved. Always reads as 0.
9	FCNTHIFG	R	0h	Start fault counter interrupt flag HFXT. If this bit is set, the FCNTIFG flag is also set. 0b = Start counter not expired. 1b = Start counter expired.
8	FCNTLFIFG	R	0h	Start fault counter interrupt flag LFXT. If this bit is set, the FCNTIFG flag is also set. 0b = Start counter not expired. 1b = Start counter expired.
7	Reserved	R	0h	Reserved. Always reads as 0.
6	DCOR_OPNIFG	R	0h	DCO external resistor open circuit fault flag. DCOR_OPNIFG can be cleared via software by CLR_DCORIFG. If the fault condition still remains, DCOR_OPNIFG is set again. 0b = DCO external resistor present 1b = DCO external resistor open circuit fault
5	DCOR_SHTIFG	R	0h	DCO external resistor short circuit fault flag. DCOR_SHTIFG can be cleared via software by CLR bit in the RSTCTL_CSRESET_CLR register. If the fault condition still remains, DCOR_SHTIFG is set again. 0b = DCO external resistor present 1b = DCO external resistor short circuit fault
4	Reserved	R	0h	Reserved. Always reads as 0.
3	Reserved	R	0h	Reserved. Always reads as 0.
2	Reserved	R	0h	Reserved. Always reads as 0.
1	HFXTIFG	R	0h	HFXT oscillator fault flag. HFXTIFG is set if a HFXT fault condition exists. HFXTIFG can be cleared via software by CLR_HFXTIFG. If the HFXT fault condition still remains, HFXTIFG is set again. 0b = No fault condition occurred after the last reset. 1b = HFXT fault. A HFXT fault occurred after the last reset.
0	LFXTIFG	R	1h	LFXT oscillator fault flag. LFXTIFG is set if a LFXT fault condition exists. LFXTIFG can be cleared via software by CLR_LFXTIFG. If the LFXT fault condition still remains, LFXTIFG is set again. 0b = No fault condition occurred after the last reset. 1b = LFXT fault. A LFXT fault occurred after the last reset.

### 5.3.10 CSCLRIFG Register (offset = 50h) [reset = 0000\_0000h]

Clock System Clear Interrupt Flag Register

**Figure 5-14. CSCLRIFG Register**

31	30	29	28	27	26	25	24
Reserved							
w1	w1	w1	w1	w1	w1	w1	w1
23	22	21	20	19	18	17	16
Reserved							
w1	w1	w1	w1	w1	w1	w1	w1
15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	CLR_FCNTHFIFG	CLR_FCNLTFIFG
w1	w1	w1	w1	w1	w1	w1	w1
7	6	5	4	3	2	1	0
Reserved	CLR_DCOR_OPNIFG	Reserved	Reserved	Reserved	Reserved	CLR_HFXTIFG	CLR_LFXTIFG
w1	w1	w1	w1	w1	w1	w1	w1

**Table 5-12. CSCLRIFG Register Description**

Bit	Field	Type	Reset	Description
31-10	Reserved	W	0h	Reserved. Always reads as 0.
9	CLR_FCNTHFIFG	W	0h	Start fault counter clear interrupt flag HFXT. Does not clear FCNTIFG. 0b = No effect 1b = Clear pending interrupt flag
8	CLR_FCNLTFIFG	W	0h	Start fault counter clear interrupt flag LFXT. Does not clear FCNTIFG. 0b = No effect 1b = Clear pending interrupt flag
7	Reserved	W	0h	Reserved. Always reads as 0.
6	CLR_DCOR_OPNIFG	W	0h	Clear DCO external resistor open circuit fault interrupt flag. 0b = No effect 1b = Clear pending interrupt flag
5	Reserved	W	0h	Reserved. Always read as 0.
4	Reserved	W	0h	Reserved. Always read as 0.
3	Reserved	W	0h	Reserved. Always read as 0.
2	Reserved	W	0h	Reserved. Always reads as 0.
1	CLR_HFXTIFG	W	0h	Clear HFXT oscillator fault interrupt flag. 0b = No effect 1b = Clear pending interrupt flag
0	CLR_LFXTIFG	W	0h	Clear LFXT oscillator fault interrupt flag. 0b = No effect 1b = Clear pending interrupt flag

### 5.3.11 CSSETIFG Register (offset = 58h) [reset = 0000\_0000h]

Clock System Clear Interrupt Flag Register

CSSETIFG is shown in [Figure 5-15](#) and described in [Table 5-13](#).

**Figure 5-15. CSSETIFG Register**

31	30	29	28	27	26	25	24
Reserved							
w1	w1	w1	w1	w1	w1	w1	w1
Reserved							
23	22	21	20	19	18	17	16
Reserved							
w1	w1	w1	w1	w1	w1	w1	w1
15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	SET_FCNTHFIFG	SET_FCNTLFIG
w1	w1	w1	w1	w1	w1	w1	w1
7	6	5	4	3	2	1	0
Reserved	SET_DCOR_OPNIFG	Reserved	Reserved	Reserved	Reserved	SET_HFXTIFG	SET_LFXTIFG
w1	w1	w1	w1	w1	w1	w1	w1

**Table 5-13. CSSETIFG Register Description**

Bit	Field	Type	Reset	Description
31-10	Reserved	W	0h	Reserved. Always reads as 0.
9	SET_FCNTHFIFG	W	0h	Start fault counter set interrupt flag HFXT. 0b = No effect 1b = Set pending interrupt flag
8	SET_FCNTLFIG	W	0h	Start fault counter set interrupt flag LFXT. 0b = No effect 1b = Set pending interrupt flag
7	Reserved	W	0h	Reserved. Always reads as 0.
6	SET_DCOR_OPNIFG	W	0h	Set DCO external resistor open circuit fault interrupt flag. 0b = No effect 1b = Set pending interrupt flag
5	Reserved	W	0h	Reserved. Always reads as 0.
4	Reserved	W	0h	Reserved. Always reads as 0.
3	Reserved	W	0h	Reserved. Always reads as 0.
2	Reserved	W	0h	Reserved. Always reads as 0.
1	SET_HFXTIFG	W	0h	Set HFXT oscillator fault interrupt flag. 0b = No effect 1b = Set pending interrupt flag
0	SET_LFXTIFG	W	0h	Set LFXT oscillator fault interrupt flag. 0b = No effect 1b = Set pending interrupt flag

### 5.3.12 CSDCOERCAL0 Register (offset = 60h) [reset = 0100\_0000h]

DCO External Resistor Calibration 0 Register

This register is used to calibrate the DCO frequency in external resistor mode for DCO frequency range (DCORSEL) 0 to 4.

**Figure 5-16. CSDCOERCAL0 Register**

31	30	29	28	27	26	25	24
Reserved						DCO_FCAL_RSEL04	
r-0	r-0	r-0	r-0	r-0	r-0	rw-<0>	rw-<1>
23	22	21	20	19	18	17	16
DCO_FCAL_RSEL04							
rw-<0>	rw-<0>	rw-<0>	rw-<0>	rw-<0>	rw-<0>	rw-<0>	rw-<0>
15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved						DCO_TCCAL	
r-0	r-0	r-0	r-0	r-0	r-0	rw-<0>	rw-<0>

**Table 5-14. CSDCOERCAL0 Register Description**

Bit	Field	Type	Reset	Description
31-26	Reserved	R	0h	Reserved. Always reads as 0.
25-16	DCO_FCAL_RSEL04	RW	100h	DCO frequency calibration for DCO frequency range (DCORSEL) 0 to 4.
15-2	Reserved	R	0h	Reserved. Always reads as 0.
1-0	DCO_TCCAL	RW	0h	DCO Temperature compensation calibration.

### 5.3.13 CSDCOERCAL1 Register (offset = 64h) [reset = 0000\_0100h]

DCO External Resistor Calibration 1 Register

This register is used to calibrate the DCO frequency in external resistor mode for DCO frequency range (DCORSEL) 5.

**Figure 5-17. CSDCOERCAL1 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
23	22	21	20	19	18	17	16
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
Reserved						DCO_FCAL_RSEL5	
r-0	r-0	r-0	r-0	r-0	r-0	rw-<0>	rw-<1>
7	6	5	4	3	2	1	0
DCO_FCAL_RSEL5							
rw-<0>	rw-<0>	rw-<0>	rw-<0>	rw-<0>	rw-<0>	rw-<0>	rw-<0>

**Table 5-15. CSDCOERCAL1 Register Description**

Bit	Field	Type	Reset	Description
31-10	Reserved	R	0h	Reserved. Always reads as 0.
9-0	DCO_FCAL_RSEL5	RW	100h	DCO frequency calibration for DCO frequency range (DCORSEL) 5.

## **Power Supply System (PSS)**

This chapter describes the operation of the Power Supply System (PSS).

Topic	Page
6.1 Power Supply System (PSS) Introduction .....	336
6.2 PSS Operation.....	337
6.3 PSS Registers .....	340

## 6.1 Power Supply System (PSS) Introduction

PSS features include:

- Wide supply voltage range: 1.62 V to 3.7 V. 1.65 V required at start-up.
- Detection of power on or off condition through VCCDET
- Generation of voltage for the device core ( $V_{CORE}$ )
- Supply voltage supervisor and monitor (SVSMH) for  $V_{CC}$
- Software accessible power-fail indicators available via the Reset Controller registers

The PSS manages all functions related to the power supply and its supervision for the device. Its primary functions are to generate a supply voltage for the core logic and to provide mechanisms for the supervision of the voltage applied to the device ( $V_{CC}$ ).

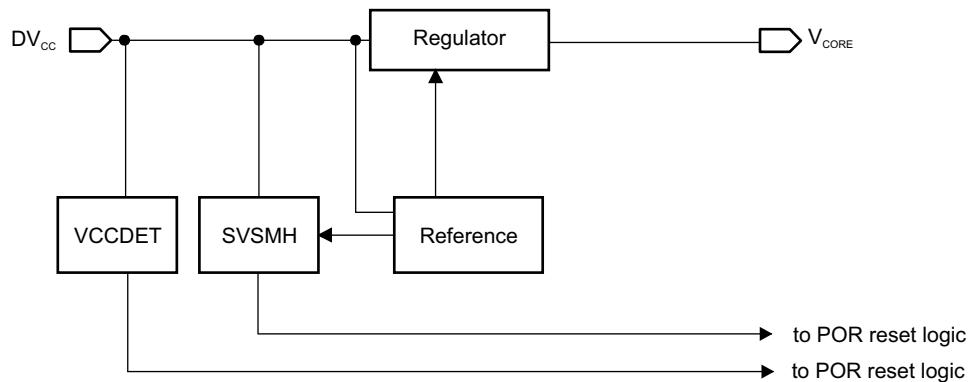
The PSS uses an integrated voltage regulator to produce a secondary core voltage ( $V_{CORE}$ ) from the primary voltage that is applied to the device ( $V_{CC}$ ). In general,  $V_{CORE}$  supplies the CPU, memories, and the digital modules, while  $V_{CC}$  supplies the I/Os and analog modules. On certain devices there are one or more separate  $AV_{CC}$  pins to supply the analog modules. However it is assumed that  $DV_{CC}$  and  $AV_{CC}$  are shorted on the board or generated from the same source, and no level shifting or isolation is done between these two supplies.

The  $V_{CORE}$  output is maintained using a dedicated voltage reference.  $V_{CORE}$  voltage level is programmable to allow power savings if the maximum device speed is not required. See the device-specific data sheet to determine if multiple core voltages are supported and the maximum system frequencies supported at each core voltage, as well as the minimum supply voltage for each core voltage. The core voltage regulators are covered in detail in the Power Control Manager (PCM) chapter.

The PSS module provides a means for  $V_{CC}$  to be supervised. The supervisor detects when the voltage falls below a specific threshold and triggers a reset or an interrupt event. The input or primary side of the regulator is referred to in this chapter as the high side.  $V_{CC}$  is supervised and monitored by the high-side supervisor/monitor (SVSMH). The thresholds enforced by SVSMH is derived from the same voltage reference used by the regulator to generate  $V_{CORE}$ .

The internal regulator and circuitry are specified by design to keep the core voltage above which timing requirements are met. What is not ensured is any event on the external  $V_{CORE}$  pin which could drop the  $V_{CORE}$  voltage such as a short circuit condition. So the user must ensure the external  $V_{CORE}$  pin is not disturbed during the device operation.

Figure 6-1 shows a representative block diagram of the PSS.



**Figure 6-1. PSS Block Diagram**

## 6.2 PSS Operation

### 6.2.1 Supply Voltage Supervisor / Monitor

The SVSMH is capable of either supervising or monitoring the high side voltage,  $V_{CC}$ . The SVSMH can be disabled using the SVSMHOFF bit to save power if the functionality is not needed. By default, the SVSMH is enabled in full performance mode. In low power modes the SVSMH can also be taken out of full performance mode by setting the SVSMHLP bit to 1 resulting in lower power but slower response time. The SVSMHLP bit is effective only in LPM3, LPM4, LPM3.5, and LPM4.5 modes.

#### 6.2.1.1 SVSMH

By default SVSMH is configured as a supervisor ( $SVSMHS = 0$ ). When configured as a supervisor and the voltage falls below the threshold the module sets the reset source bit in the reset controller register and asserts a reset. When SVSMH is configured as a monitor and the  $V_{CC}$  voltage falls below the SVSMH threshold the module sets the SVSMHIFG interrupt flag and generates an interrupt (only if  $SVSMHIE = 1$ ). If the voltage remains below the SVSMH level and software attempts to clear SVSMHIFG, it is immediately set again by hardware. The interrupt flag of SVSMH remains set until cleared by a hard reset or by software.

When the SVSMH is enabled by clearing a previously set SVSMHOFF bit, a delay element masks the interrupt and reset sources until the SVSMH circuit has settled as indicated in the SVSMH on/off delay time in the device-specific data sheet. The SVSMH module has configurable performance modes for power-saving operation. See the SVSMH electrical specification in the device specific data sheet for information on current consumption and response time of SVSMH in full performance and low power normal performance modes.

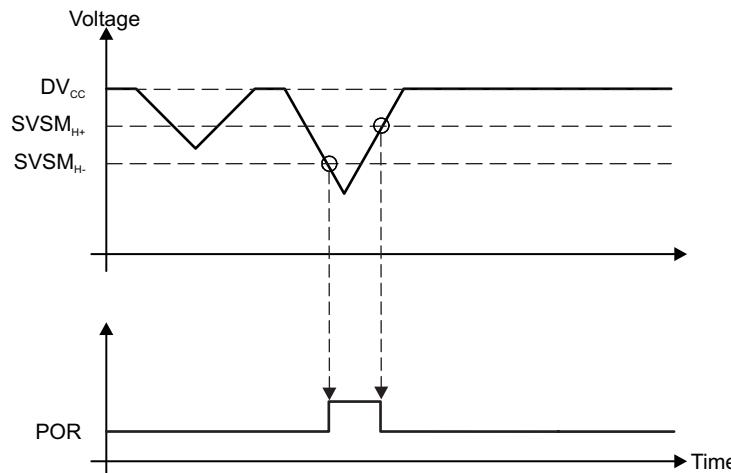
#### 6.2.1.2 SVMHOUT

When SVSMH is configured in monitor mode ( $SVSMHS = 1$ ) the SVMHOE bit selects the output enable for SVSMH. When the output is enabled the SVSMHIFG bit value is output on the SVMHOUT device pin. The device specific port logic must be configured accordingly. The SVMHOUTPOLAL bit determines if SVMHOUT is active low or active high. When the SVMHOUT functionality is disabled through clearing SVMHOE bit, the SVMHOUT pin always reflects the status corresponding to healthy  $DV_{CC}$ .

#### 6.2.1.3 SVSMH Thresholds

The SVSMHTH bits define the voltage threshold for monitoring/supervising  $V_{CC}$ . These settings should be selected according to the minimum voltages required for device operation in a given application, as well as system power supply characteristics. See the device specific data sheet for threshold values. As [Figure 6-2](#) shows, there is hysteresis built into the supervision thresholds, such that the thresholds in force depend on whether the voltage rail is going up or down.

The behavior of the SVSMH according to these thresholds is best represented graphically. [Figure 6-2](#) shows how the supervisor responds to supply failure conditions.



**Figure 6-2. Supply Voltage Failure and Resulting PSS Action**

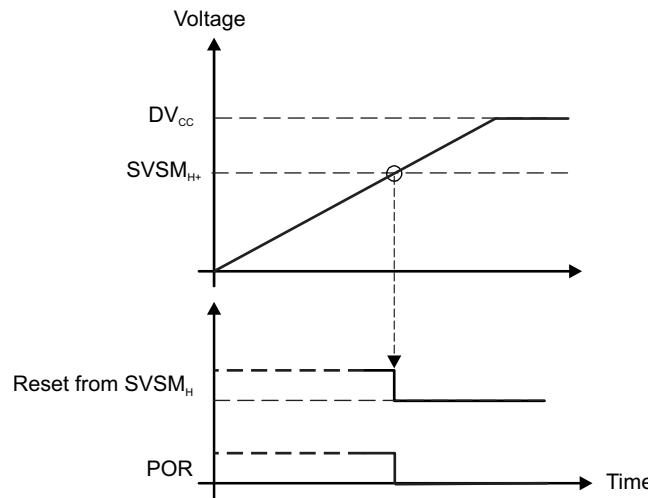
---

**NOTE:** If the SVSMH is set to supervisor mode, and supply falls below the user-defined threshold level as per the [PSSCTL0 Register](#), the POR issued clears the threshold setting and the device subsequently wakes up at the default start-up  $V_{CC}$  threshold voltage of 1.65 V.

---

### 6.2.2 Supply Voltage Supervisor during Power-Up

When the device is powering up, the SVSMH function is enabled by default as supervisor. Initially,  $V_{CC}$  is low, and therefore the PSS holds the device in POR reset. Once the SVSMH level is met, the reset is released. [Figure 6-3](#) shows the power-up process.



**Figure 6-3. PSS Action at Device Power-Up**

### 6.2.3 VCCDET

VCCDET or VCC Detect is an extremely low power circuit primarily intended for proper and reliable power-up or power-down of the device. As the VCCDET is extremely low power its threshold varies over a large voltage range. See the device specific data sheet for VCCDET threshold voltages. VCCDET is always active including the device operation in low power modes.

Because the threshold voltage of VCCDET is not precise, it is not recommended to use VCCDET for monitoring the supply voltage on-chip. Applications that require precise monitoring of supply voltage on-chip should rather use SVSMH module. See the device specific data sheet for the SVSMH threshold voltages.

In case of supply failure when SVSMH is disabled by the application or in case of a sudden failure of supply at a rate faster than recommended for SVSMH module, the VCCDET generates a POR reset. As described earlier, this does not ensure a controlled power-down of the device, rather an asynchronous reset. When the supply is available subsequently, the device goes through the power-up sequence and reaches default active mode once the POR is de-asserted.

#### 6.2.4 PSS Interrupts

There is only one interrupt condition from PSS that is triggered by SVSMH module while operating in the monitor mode. The interrupt condition is available through SVSMHIFG bit. When SVSMHIE bit is set to 1, the interrupt is asserted from PSS. This interrupt can be handled as maskable or non-maskable interrupt by configuring the respective SYSCTL and NVIC registers.

## 6.3 PSS Registers

The PSS registers are listed in [Table 6-1](#). The base address of the PSS module can be found in the device-specific data sheet. The address offset of each PSS register is given in [Table 6-1](#).

**Table 6-1. PSS Registers**

Offset	Acronym	Register Name	Section
00h	PSSKEY	Key Register	<a href="#">Section 6.3.1</a>
04h	PSSCTL0	Control 0 Register	<a href="#">Section 6.3.2</a>
34h	PSSIE	Interrupt Enable Register	<a href="#">Section 6.3.3</a>
38h	PSSIIFG	Interrupt Flag Register	<a href="#">Section 6.3.4</a>
3Ch	PSSCLRIFG	Clear Interrupt Flag Register	<a href="#">Section 6.3.5</a>

---

**NOTE:** This is a 32-bit module and can be accessed through word (32-bit) or half-word (16-bit) or byte (8-bit) accesses.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### **6.3.1 PSSKEY Register (offset = 00h) [reset = 0000A596h]**

PSS Key Register

**Figure 6-4. PSSKEY Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
PSSKEY							
rw-1	rw-0	rw-1	rw-0	rw-0	rw-1	rw-0	rw-1
7	6	5	4	3	2	1	0
PSSKEY							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0

**Table 6-2. PSSKEY Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always read 0.
15-0	PSSKEY	RW	A596h	PSS key. Always read as A596h. Must be written with 695Ah to unlock the PSS registers for writing. Any other write value locks the PSS registers. Note: Registers can be read even when locked.

### 6.3.2 PSSCTL0 Register (offset = 04h) [reset = 00002000h]

PSS Control 0 Register

**Figure 6-5. PSSCTL0 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
Reserved		Reserved		Reserved	DCDC_FORCE	Reserved	
r-0	r-0	rw-1	rw-0	r-0	rw-0	r-0	r-0
7	6	5	4	3	2	1	0
SVMHOUTPOLAL	SVMHOE	SVSMHHTH			SVSMHS	SVSMHLP	SVSMHOFF
rw-0	rw-0	rw-0 <sup>(1)</sup>	rw-0 <sup>(1)</sup>	rw-0 <sup>(1)</sup>	rw-0	rw-0 <sup>(1)</sup>	rw-0 <sup>(1)</sup>

<sup>(1)</sup> This bit is reset to 0 when the device enters LPM3.5 or LPM4.5 modes. If this bit is set to 1 when the device enters LPM3.5 or LPM4.5 modes, the functionality associated with the bit value 1 is retained through the LPM3.5 or LPM4.5 modes.

**Table 6-3. PSSCTL0 Register Description**

Bit	Field	Type	Reset	Description
31-14	Reserved	R	0h	Reserved. Always reads as 0.
13-12	Reserved	RW	2h	Internal configuration. Changing this may cause device to reset during core voltage level transitions.
11	Reserved	R	0h	Reserved. Always reads as 0.
10	DCDC_FORCE	RW	0h	Force DC-DC regulator operation. Refer to Power Control Manager (PCM) chapter for details about this feature. 0b = DC-DC regulator operation not forced. Automatic fail-safe mechanism switches the core voltage regulator from DC-DC to LDO when the supply voltage falls below the minimum supply voltage necessary for DC-DC operation. 1b = DC-DC regulator operation forced. Automatic fail-safe mechanism is disabled and device continues to operate out of DC-DC regulator.
9-8	Reserved	R	0h	Reserved. Always reads as 0.
7	SVMHOUTPOLAL	RW	0h	SVMHOUT pin polarity active low. 0b = SVMHOUT is active high. An error condition is signaled by a 1 at the SVMHOUT pin. 1b = SVMHOUT is active low. An error condition is signaled by a 0 at the SVMHOUT pin.
6	SVMHOE	RW	0h	SVSM high-side output enable 0b = SVSMHIFG bit is not output. 1b = SVSMHIFG bit is output to the device SVMHOUT pin. The device-specific port logic must be configured accordingly.
5-3	SVSMHHTH	RW	0h	SVSM high-side reset voltage level. If DVCC falls short of the SVSMH voltage level selected by SVSMHHTH, a reset is triggered (if SVSMHOFF = 0 and SVSMHS = 0) or interrupt is triggered (if SVSMHOFF = 0 and SVSMHS = 1). The voltage levels are defined in the device-specific data sheet.
2	SVSMHS	RW	0h	Supply supervisor or monitor selection for the high-side 0b = Configure as SVSH 1b = Configure as SVMH
1	SVSMHLP	RW	0h	SVSM high-side low power normal performance mode 0b = Full performance mode. See the device-specific data sheet for response times. 1b = Low power normal performance mode in LPM3, LPM4, and LPMx.5, full performance in all other modes. See the device-specific data sheet for response times.

**Table 6-3. PSSCTL0 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
0	SVSMHOFF	RW	0h	<p>SVSM high-side off            0b = The SVSMH is on.            1b = The SVSMH is off.</p> <p>Note: If the SVSMH is kept disabled in Active Mode, and is enabled before entering a low power mode of the device (LPM3/LPM4/LPMx.5) care should be taken that sufficient time has elapsed since enabling of the module until entry into the device low power mode to allow for successful wakeup of SVSMH module as per 'SVSMH on/off delay time' spec in respective device datasheet. Otherwise, SVSMH may trip, causing device to get a Reset and wakeup from the Low Power Mode.</p> <p>Note: If the SVSMH is disabled when in LPM4.5, and the supply goes below default SVSMH limits, the device may get an SVSMH-triggered Reset if woken up while the supply is low.</p> <p>Note: If SVSMH is disabled by setting this bit, and immediately enabled due to RSTn pin reset within 200ns, the reset source may get elevated to a Reset High side due to insufficient power down time allowed for SVSMH.</p>

### 6.3.3 PSSIE Register (offset = 34h) [reset = 0000h]

PSS Interrupt Enable Register

**Figure 6-6. PSSIE Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved						SVSMHIE	Reserved
r0	r0	r0	r0	r0	r0	rw-0	r-0

**Table 6-4. PSSIE Register Description**

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0h	Reserved. Always read 0.
1	SVSMHIE	RW	0h	High-side SVSM interrupt enable, when set as a monitor (SVSMHS = 1). 0b = Interrupt disabled 1b = Interrupt enabled Make sure that the SVSMHIFG bit is cleared before enabling interrupt. Otherwise an unexpected NMI may be seen due to an earlier dip in DVCC while interrupt was disabled.
0	Reserved	R	0h	Reserved. Always read 0.

### **6.3.4 PSSIFG Register (offset = 38h) [reset = 0000h]**

PSS Interrupt Flag Register

**Figure 6-7. PSSIFG Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved						SVSMHIFG	Reserved
r0	r0	r0	r0	r0	r-0	r-0	r-0

**Table 6-5. PSSIFG Register Description**

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0h	Reserved. Always read 0.
1	SVSMHIFG	R	0h	<p>High-side SVSM interrupt flag.            SVSMH = 0 (supervisor mode): The SVSMHIFG interrupt flag is not active.            SVSMH = 1 (monitor mode): The SVSMHIFG interrupt flag is set if DVCC drops below the SVSMH power-down level and an interrupt is generated. The bit is cleared by software.</p> <p>0b = No interrupt pending            1b = Interrupt due to SVSMH</p> <p>The interrupts generated by the PSS can be classified either as NMI or regular interrupts at the device level. For more details, refer to the system control section in the appropriate device datasheet.</p>
0	Reserved	R	0h	Reserved. Always read 0.

### 6.3.5 PSSCLRIFG Register (offset = 3Ch) [reset = 0000h]

PSS Clear Interrupt Flag Register

**Figure 6-8. PSSCLRIFG Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved						CLRSVSMHIF G	Reserved
r0	r0	r0	r0	r0	r0	w-0	r-0

**Table 6-6. PSSCLRIFG Register Description**

Bit	Field	Type	Reset	Description
31-2	Reserved	R	0h	Reserved. Always read 0.
1	CLRSVSMHIFG	W	0h	SVSMH clear interrupt flag 0b = No effect 1b = Clear pending interrupt flag
0	Reserved	R	0h	Reserved. Always read 0.

## **Power Control Manager (PCM)**

---



---



---

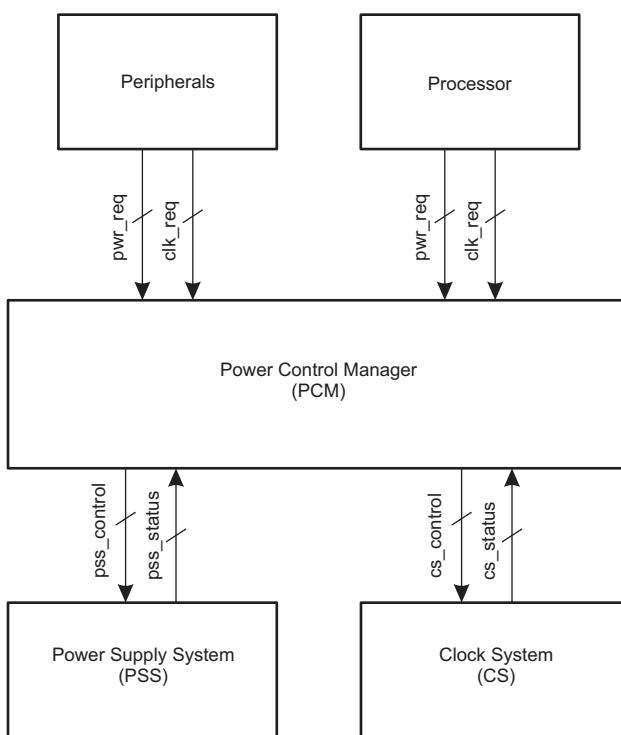
Topic	Page
7.1 PCM Introduction.....	348
7.2 PCM Overview.....	348
7.3 Core Voltage Regulators .....	349
7.4 Power Modes .....	349
7.5 Power Mode Transitions.....	353
7.6 Changing Core Voltages.....	356
7.7 ARM Cortex Processor Sleep Modes.....	357
7.8 Power Mode Requests .....	358
7.9 Power Mode Selection.....	358
7.10 Power Mode Transition Checks .....	359
7.11 Power Mode Clock Checks .....	359
7.12 Clock Configuration Changes .....	360
7.13 Changing Active Modes .....	360
7.14 Entering LPM0 Modes .....	362
7.15 Exiting LPM0 Modes .....	362
7.16 Entering LPM3 or LPM4 Modes .....	362
7.17 Exiting LPM3 or LPM4 Modes .....	363
7.18 Entering LPM3.5 and LPM4.5 Modes .....	363
7.19 Exiting LPM3.5 and LPM4.5 Modes .....	364
7.20 Supply Voltage Supervisor and Monitor and Power Modes.....	365
7.21 Low-Power Reset.....	366
7.22 Power Requests During Debug .....	366
7.23 Wake-up Sources From Low Power Modes.....	367
7.24 PCM Registers .....	368

## 7.1 PCM Introduction

MSP432P4xx family of devices supports several power modes that allow for the optimization of power for a given application scenario. The power modes can be changed dynamically to cover many different power profile requirements across many applications. The Power Control Manager (PCM) is responsible for managing power requests from different areas of the system and processing the requests in a controlled manner. It uses all information from the system that may affect the power requirements and adjusts the power as required, if possible. The Clock System (CS) and the Power Supply System (PSS) settings are the two primary elements that control the power settings of the device and hence the power consumption of the device.

## 7.2 PCM Overview

There are several factors that determine the power mode setting of the device. This includes existing conditions from the clock system (CS) settings and the power supply system (PSS) settings. It is possible that a power mode request cannot be safely entered based on existing conditions in the system. The PCM is an automated subsystem that adjusts power based on direct power request settings or indirectly based on other requests in the system. The PCM becomes the main interface between the PSS and CS modules as shown in [Figure 7-1](#).



**Figure 7-1. Power Control Manager Interaction**

The PCM reacts based on events. The most common events are:

- PCM Control 0 Register (PCMCTL0). This register can be modified directly by the application execution to request that a particular power mode be entered.
- Interrupt and wake-up events. Interrupts from low-power modes cause operation to automatically return to an active mode.
- Reset events. Reset events cause the power mode to be set back to its default setting.
- Debug events. Power mode settings are adapted to support debug hardware requirements.

All of these events, regardless of their source, cause a power change request to the PCM. The power change request is used as an indication that a new power mode setting may be required. The PCM processes all requests and makes the necessary changes to the system as required. It may not be possible to fulfill all requests and the PCM may deny some requests.

## 7.3 Core Voltage Regulators

$DV_{CC}$  can be powered from a wide input voltage range, but the core logic of the device must be kept at a voltage lower than what this range allows. For this reason, a linear regulator (alternately referred to as LDO) has been integrated into the PSS. The regulator derives the necessary core voltage ( $V_{CORE}$ ) from  $DV_{CC}$ .

On devices which support multiple core voltage levels higher MCLK speeds require higher  $V_{CORE}$  voltages. If a lower  $V_{CORE}$  voltage supports the frequency needed it should be selected for the lowest power consumption. The core voltage level is controlled through the power mode selection. Note that the default setting, the lowest core voltage, enables operation of MCLK over a very wide frequency range. As such, no changes are required for many applications. See the device-specific data sheet for performance characteristics, core voltage levels supported and any minimum  $DV_{CC}$  restrictions for each of the core voltage levels.

Before increasing MCLK to a higher speed, it is necessary for software to ensure that the  $V_{CORE}$  level is sufficiently high for the chosen frequency. Failure to do so may force the CPU to attempt operation without sufficient power, which can cause unpredictable results or issue a POR reset. See [Section 7.6](#) for more details.

The regulator supports different load settings to optimize power. The hardware controls the load settings automatically, according to the following criteria:

- Which power mode is selected
- Whether or not JTAG is active

Some devices offer an inductor based DC-DC regulator in addition to a LDO to regulate the core voltage. See the device-specific data sheet for the availability of DC-DC regulator on the device. The DC-DC provides increased efficiency and therefore decreased current consumption from  $DV_{CC}$ , however it requires an inductor to be connected on the  $V_{SW}$  pin as well as a larger capacitor on the  $V_{CORE}$  pin. The larger capacitor results in longer wake-up times when the core voltage is changed and at startup.

The device always starts up with the LDO and then the user can switch to the DC-DC if so desired by choosing a power mode that enables the DC-DC. In general, the larger the supply voltage the more current savings with the DC-DC. The DC-DC efficiency is also based on load currents, and it is more efficient with higher currents. See the device-specific data sheet for DC-DC electrical specification.

### 7.3.1 DC-DC Regulator Care Abouts

The proper size of decoupling capacitor and inductor as recommended in the datasheet must be chosen. The DC-DC operation automatically switches to the LDO when  $V_{CC}$  drops too low for the DC-DC to remain more efficient than the LDO. This automatic switchover can be disabled by the user by setting the DCDC\_FORCE bit in the [PSSCTL0 Register](#). Refer to the device-specific data sheet for the switchover voltage.

---

**NOTE: Caution:** If the components specified for reliable DC-DC operation are not connected, or their values are outside the allowed range, and software attempts to enable the DC-DC regulator causes a POR reset to be issued or result in indeterministic device behavior.

---

## 7.4 Power Modes

Each device supports several power modes. Active modes (AM) are any power mode in which CPU execution is possible. LPM0, LPM3, LPM4, and LPMx.5 modes do not allow for CPU execution. Each power mode is described in the following sections. Direct transitions between some of the power modes are not possible - see [Section 7.5](#) for details on supported transitions.

#### 7.4.1 Active Modes (AM)

Active modes refer to any power mode in which CPU execution is possible. There are six active modes available. Active modes can be logically grouped by the core voltage levels that are supported. There are two core voltage level settings: core voltage level 0 and core voltage level 1. Three active modes are associated with each core voltage level. The various active modes allow for optimal power and performance across a broad range of application requirements. The core voltage can be supplied either by a low dropout (LDO) regulator or a DC-DC regulator.

AM\_LDO\_VCORE0 is based on the core voltage level 0 with LDO operation. Similarly, AM\_LDO\_VCORE1 is based on the core voltage level 1 with LDO operation. Each of these active modes supports a different maximum frequency of operation. The higher the core level setting, the higher the maximum frequency supported. As shown in [Figure 7-3](#), it is only possible to change core voltage levels by transitioning to and from AM\_LDO\_VCORE0 and AM\_LDO\_VCORE1.

AM\_DCDC\_VCORE0 allows operation using the integrated DC-DC regulator at core voltage level 0. The DC-DC regulator is useful if long periods of activity are required. The efficiency of the DC-DC regulator depends on the VCC voltage level, the core voltage level, and the frequency of operation.

AM\_DCDC\_VCORE1 is the same as AM\_DCDC\_VCORE0, except that the core voltage level setting is increased and can support a higher frequency of operation.

For AM\_LDO\_VCORE0 and AM\_DCDC\_VCORE0 modes, the maximum CPU operating frequency is 24 MHz and maximum input clock frequency for peripherals is 12 MHz. For AM\_LDO\_VCORE1 and AM\_DCDC\_VCORE1 modes, the maximum CPU operating frequency is 48 MHz and maximum input clock frequency for peripherals is 24 MHz.

AM\_LF\_VCORE0 and AM\_LF\_VCORE1 are the active modes useful for low frequency operation. These low-frequency active modes constrain the maximum frequency of operation to 128 kHz. Also, **the application must not perform flash program/erase operations or any change to SRAM bank enable or retention enable configurations while in low-frequency active modes**. This allows the power management system to be optimized to support the lower power demands in these operating modes. Low-Frequency active modes are based on LDO operation only.

To use the DC-DC regulator, the application must always transition first to either AM\_LDO\_VCORE0 or AM\_LDO\_VCORE1. Because the DC-DC regulator is a switching regulator, it takes time to settle and achieve a stable voltage. See the device specific datasheet for AM\_LDO\_VCOREx to AM\_DCDC\_VCOREx transition latencies. During this settling time, the LDO automatically remains enabled. When the DC-DC regulator has settled, the LDO is automatically disabled to reduce power.

All active modes support code execution from flash memory or SRAM. Note that not all active mode transitions are supported. See [Figure 7-3](#) for all valid active mode transitions. Active modes are referred as Run modes in ARM terminology.

#### 7.4.2 LPM0

During LPM0, the processor execution is halted. Halting the processor reduces dynamic power due to reduced switching activities caused by execution of the processor. In general, there is only one LPM0 setting, and LPM0 can be entered from all active modes. Therefore, LPM0 effectively supports six different modes of operation, corresponding to each active mode (LPM0\_LDO\_VCORE0, LPM0\_DCDC\_VCORE0, LPM0\_LF\_VCORE0, LPM0\_LDO\_VCORE1, LPM0\_DCDC\_VCORE1, and LPM0\_LF\_VCORE1). The maximum frequency requirement in LPM0 is identical to that of the active mode at the time of LPM0 entry. For example, if LPM0 is entered from AM\_LDO\_VCORE0, the maximum frequency requirement of AM\_LDO\_VCORE0 would also apply during LPM0\_LDO\_VCORE0. LPM0 is useful to save power when processor execution is not required, yet very fast wake up time is necessary. LPM0 exit always takes the device back to the original active mode at the time of LPM0 entry. See [Figure 7-4](#) for all valid LPM0 transitions. LPM0 modes are referred as Sleep modes in ARM terminology.

#### 7.4.3 LPM3 and LPM4

Similar to LPM0, the processor execution is halted during LPM3 or LPM4. LPM3 mode restricts maximum frequency of device operation to 32.768 kHz. Only RTC and WDT modules are functional out of low-frequency clock sources (LFXT, REFO, and VLO) while in LPM3. All other peripherals are disabled and high-frequency clock sources are turned off in LPM3. LPM4 mode is entered when the device is programmed for LPM3 with RTC and WDT modules disabled. All clock sources are turned off in LPM4 mode and no peripheral functionality is available. **All SRAM banks that are enabled for data retention during LPM3 or LPM4 and all peripheral registers retain the data through LPM3 and LPM4 modes.** The device I/O pin states are also latched and retained in LPM3 and LPM4 modes.

LPM3 and LPM4 modes are useful for relatively infrequent processor activity followed by long periods of low frequency activity, better known as low duty cycle applications. Refer to [Table 7-8](#) for LPM3 and LPM4 wake-up sources. The wake-up time from LPM3 and LPM4 is longer to that of LPM0, but the average power consumption is significantly lower. See the device specific datasheet for wake-up time from different low power modes. DC-DC regulator operation is not supported in LPM3 and LPM4 modes. See [Figure 7-5](#) for all supported LPM3 and LPM4 transitions. LPM3 or LPM4 exit always takes the device back to the original active mode at the time of LPM3 or LPM4 entry. LPM3 and LPM4 modes are referred as Deep Sleep modes in ARM terminology.

#### 7.4.4 LPM3.5 and LPM4.5

LPM3.5 and LPM4.5 modes provide the lowest power consumption possible but at reduced functionality.

In LPM3.5 mode, all peripherals are disabled and powered down except for the RTC and WDT, which can be optionally enabled by the application and clocked out of low-frequency clock sources (LFXT, REFO, and VLO). Furthermore, the device is brought down to the core voltage level 0. Wake up is possible through any of the wake-up events mentioned in [Table 7-8](#). **LPM3.5 mode does not retain any peripheral register data.** In LPM3.5 mode however, **Bank-0 of SRAM is retained for application's use as a backup memory and also device I/O pin states are latched and retained.**

In LPM4.5 mode, all peripherals and clock sources are powered down and the internal voltage regulator is switched off. Wake up is possible through any of the wake-up events mentioned in [Table 7-8](#). **LPM4.5 mode does not retain any SRAM or peripheral register data but device I/O pin states are latched and retained.** Any essential data must be stored to flash prior to entering LPM4.5 mode.

LPM3.5 and LPM4.5 modes are useful for complete power down or simple time keeping over long periods of time with infrequent wake-up activity. See [Section 7.5.4](#) for valid LPM3.5 and LPM4.5 transitions.

LPM3.5 and LPM4.5 modes are referred as Stop or Shut Down modes in ARM terminology.

#### 7.4.5 Summary of Power Modes

The summary of different power modes, their functional capabilities and application level constraints are given in [Table 7-1](#).

**Table 7-1. Power Modes Summary**

<b>Power Mode</b>	<b>Operating State</b>	<b>Features and Application Constraints</b>
Active Mode (Run Mode)	AM_LDO_VCORE0	LDO or DC-DC regulator based active modes at core voltage level 0. CPU is active and full peripheral functionality is available. CPU and DMA maximum operating frequency is 24 MHz. Peripherals maximum input clock frequency is 12 MHz. All low and high-frequency clock sources can be active. Flash memory and all enabled SRAM banks are active.
	AM_DCDC_VCORE0	
	AM_LDO_VCORE1	LDO or DC-DC regulator based active modes at core voltage level 1. CPU is active and full peripheral functionality is available. CPU and DMA maximum operating frequency is 48 MHz. Peripherals maximum input clock frequency is 24 MHz. All low and high-frequency clock sources can be active. Flash memory and all enabled SRAM banks are active.
	AM_DCDC_VCORE1	
	AM_LF_VCORE0	LDO based low-frequency active modes at core voltage level 0 or 1. CPU is active and full peripheral functionality is available. CPU, DMA and peripherals maximum operating frequency is 128 kHz. Only low-frequency clock sources (LFXT, REFO, and VLO) can be active. All high-frequency clock sources need to be disabled by application. Flash memory and all enabled SRAM banks are active.
	AM_LF_VCORE1	Flash erase/program operations and SRAM bank enable or retention enable configuration changes must not be performed by application. DC-DC regulator can not be used.
LPM0 (Sleep)	LPM0_LDO_VCORE0	LDO or DC-DC regulator based operating modes at core voltage level 0. CPU is inactive but full peripheral functionality is available.
	LPM0_DCDC_VCORE0	DMA maximum operating frequency is 24 MHz. Peripherals maximum input clock frequency is 12 MHz. All low and high-frequency clock sources can be active. Flash memory and all enabled SRAM banks are active.
	LPM0_LDO_VCORE1	LDO or DC-DC regulator based operating modes at core voltage level 1. CPU is inactive but full peripheral functionality is available.
	LPM0_DCDC_VCORE1	DMA maximum operating frequency is 48 MHz. Peripherals maximum input clock frequency is 24 MHz. All low and high-frequency clock sources can be active. Flash memory and all enabled SRAM banks are active.
	LPM0_LF_VCORE0	LDO based low-frequency operating modes at core voltage level 0 or 1. CPU is inactive but full peripheral functionality is available. DMA and peripherals maximum operating frequency is 128 kHz. Only low-frequency clock sources (LFXT, REFO, and VLO) can be active. All high-frequency clock sources need to be disabled by application. Flash memory and all enabled SRAM banks are active.
	LPM0_LF_VCORE1	Flash erase/program operations and SRAM bank enable or retention enable configuration changes must not be performed by application. DC-DC regulator can not be used.

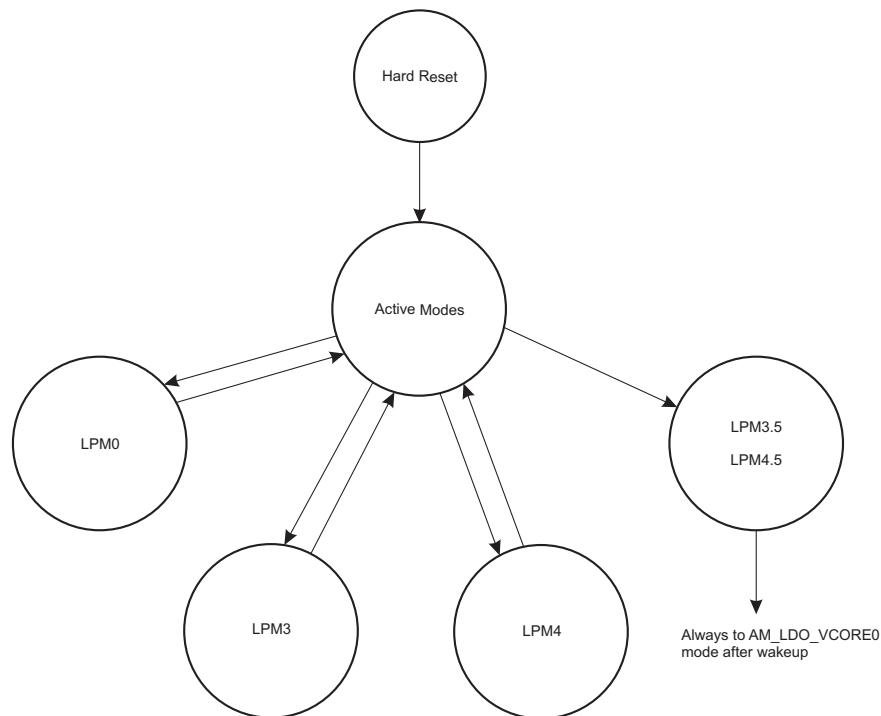
**Table 7-1. Power Modes Summary (continued)**

<b>Power Mode</b>	<b>Operating State</b>	<b>Features and Application Constraints</b>
LPM3 (Deep Sleep)	LDO_VCORE0	LDO based operating modes at core voltage level 0 or 1. CPU is inactive and peripheral functionality is reduced. Only RTC and WDT modules can be functional with maximum input clock frequency of 32.768 kHz. All other peripherals and retention enabled SRAM banks are kept under state retention power gating.
	LDO_VCORE1	Flash memory is disabled. SRAM banks not configured for retention are disabled. Only low-frequency clock sources (LFXT, REFO, and VLO) can be active. All high-frequency clock sources are disabled. Device I/O pin states are latched and retained. DC-DC regulator can not be used.
LPM4 (Deep Sleep)	LDO_VCORE0	LDO based operating modes at core voltage level 0 or 1. Achieved by entering LPM3 with RTC and WDT modules disabled. CPU is inactive with no peripheral functionality. All peripherals and retention enabled SRAM banks are kept under state retention power gating.
	LDO_VCORE1	Flash memory is disabled. SRAM banks not configured for retention are disabled. All low and high-frequency clock sources are disabled. Device I/O pin states are latched and retained. DC-DC regulator can not be used.
LPM3.5 (Stop or Shut Down)	LDO_VCORE0	LDO based operating mode at core voltage level 0. Only RTC and WDT modules can be functional with maximum input clock frequency of 32.768 kHz. CPU and all other peripherals are powered down. Only Bank-0 of SRAM is under data retention. All other SRAM banks and flash memory are powered down. Only low-frequency clock sources (LFXT, REFO, and VLO) can be active. All high-frequency clock sources are disabled. Device I/O pin states are latched and retained. DC-DC regulator can not be used.
LPM4.5 (Stop or Shut Down)	VCORE_OFF	Core voltage is turned off. CPU, flash memory, all SRAM banks, and all peripherals are powered down. All low and high-frequency clock sources are powered down. Device I/O pin states are latched and retained.

## 7.5 Power Mode Transitions

Several power mode transitions are possible under application control, allowing for optimal power performance tradeoffs for a wide variety of usage profiles. The high level representation of different power mode transitions is given in [Figure 7-2](#) and the detailed description of all supported power mode transitions is covered in the subsequent sections.

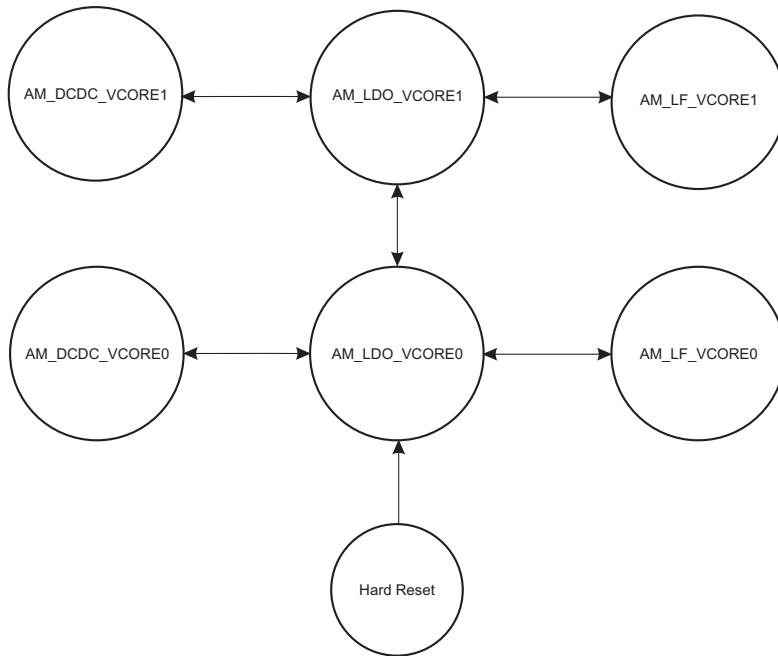
After power up or upon hard reset or any other higher class reset the device enters active mode. From active mode it is possible to transition into different low-power modes LPM0, LPM3, LPM4, LPM3.5, and LPM4.5 through application programming. Upon defined wake-up events the device returns to the active mode from which the specific low-power mode was entered. Only for LPM3.5 and LPM4.5 modes, the device always enters AM\_LDO\_VCORE0 mode after wake-up.



**Figure 7-2. High Level Power Mode Transitions**

### 7.5.1 Active Mode Transitions

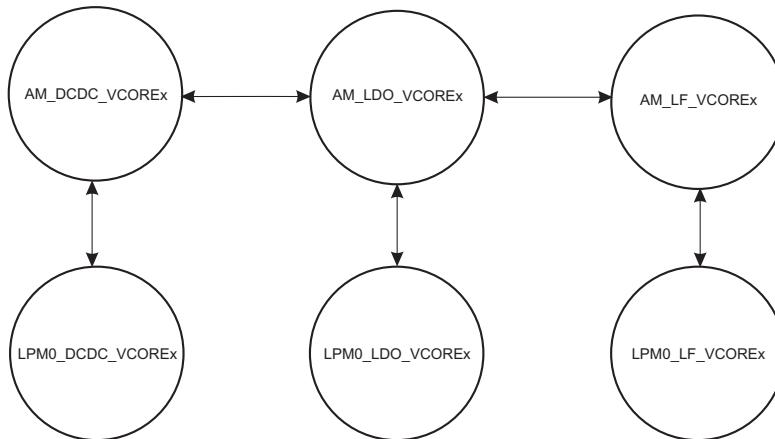
The device enters AM\_LDO\_VCORE0 mode after hard reset or any higher class of reset. It is possible to dynamically change from one active mode to another during code execution as long as the transition is valid. Not all transitions are possible. [Figure 7-3](#) shows all active mode transitions that are supported. See the device-specific data sheet for active mode transition latencies.



**Figure 7-3. Valid Active Mode Transitions**

### 7.5.2 Transitions To and From LPM0

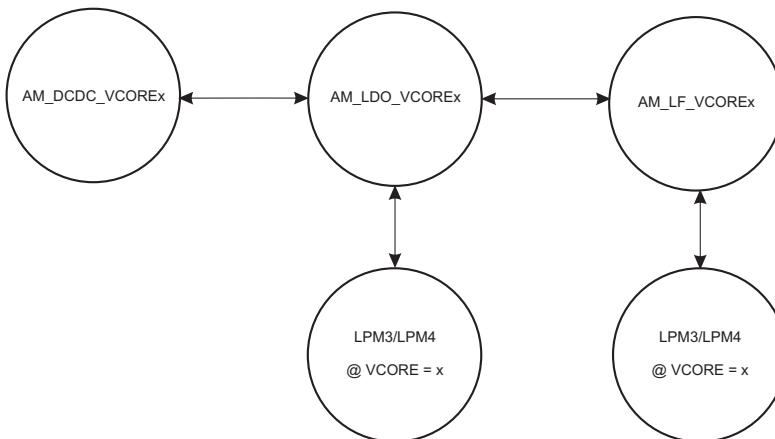
Entry to LPM0 is possible from any valid active mode. Therefore, each active mode has a corresponding LPM0 mode. LPM0 mode is always entered at the same core voltage level setting as the active mode at the time of LPM0 entry. For example, LPM0 entry from AM\_LDO\_VCORE0, which has a core voltage level 0, is also to core voltage level 0. In addition, all frequency and voltage constraints of the active mode at the time of LPM0 entry also apply to the LPM0 mode. It is not possible to switch between different LPM0 modes without first exiting the current LPM0 mode. Exiting from any LPM0 mode is always back to the respective active mode at the time of LPM0 entry. [Figure 7-4](#) shows all LPM0 mode transitions supported. See the device-specific data sheet for LPM0 mode transition latencies.



**Figure 7-4. Valid LPM0 Transitions**

### 7.5.3 Transitions To and From LPM3 and LPM4

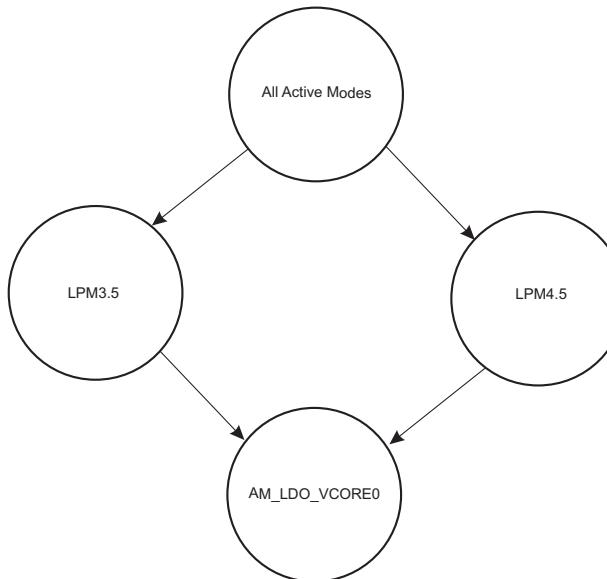
Entry to LPM3 and LPM4 modes is possible from any valid active mode, with the exception of AM\_DCDC\_VCOREx modes. The maximum frequency of operation during LPM3 mode is 32.768 kHz. LPM3 and LPM4 modes are always entered at the same core voltage level setting as the active mode at the time of LPM3 or LPM4 entry. For example, LPM3 or LPM4 entry from AM\_LDO\_VCORE0, which has a core voltage level 0, is also to core voltage level 0. The DC-DC operation must always be disabled by the application during entry to LPM3 or LPM4 modes. The application must not attempt a LPM3 or LPM4 entry from AM\_DCDC\_VCOREx modes, failure to do so results in the LPM\_INVALID\_TR\_IFG being set and the transition to be aborted by the PCM. Exiting from any LPM3 or LPM4 mode is always back to the active mode at the time of LPM3 or LPM4 entry. [Figure 7-5](#) shows all LPM3 and LPM4 mode transitions supported. See the device-specific data sheet for LPM3 and LPM4 mode transition latencies.



**Figure 7-5. Valid LPM3 and LPM4 Transitions**

#### 7.5.4 Transitions To and From LPM3.5 and LPM4.5

Entry to LPM3.5 and LPM4.5 modes is possible from any valid active mode. However, LPM3.5 and LPM4.5 mode entry is always to the lowest core voltage level setting regardless what active mode it was entered from. Exiting from LPM3.5 and LPM4.5 modes always goes back to the default active mode, AM\_LDO\_VCORE0. Entering and exiting LPM3.5 and LPM4.5 modes requires specific steps for proper entry and exit. Refer to [Section 7.18](#) and [Section 7.19](#) for details. [Figure 7-6](#) shows all LPM3.5 and LPM4.5 mode transitions supported. See the device-specific data sheet for LPM3.5 and LPM4.5 mode transition latencies.



**Figure 7-6. Valid LPM3.5 and LPM4.5 Transitions**

## 7.6 Changing Core Voltages

### 7.6.1 Increasing $V_{CORE}$ for Higher MCLK Frequencies

With a reset,  $V_{CORE}$  and SVSMH threshold, default to their lowest possible levels. These default settings allow a wide range of MCLK operation, and in many applications no change to these levels is required. However, if the application requires the performance provided by higher MCLK frequencies the  $V_{CORE}$  level must be increased. See the device specific data sheet for the specific maximum MCLK frequency at each  $V_{CORE}$  levels.

The user must first refer to the device specific data sheet to see if the new  $V_{CORE}$  level has a higher minimum  $V_{CC}$  voltage requirement and if so that it is met. Then software must request the new  $V_{CORE}$  level voltage and check that it is met before changing MCLK, because failing to supply sufficient voltage to the CPU could produce unpredictable results and / or result in a POR reset.

After requesting a power mode with a higher  $V_{CORE}$  level when the current power mode uses a lower  $V_{CORE}$  level, there is a time delay until the new voltage has been established. Software must not increase the MCLK frequency until the necessary core voltage has settled.

### 7.6.2 Decreasing $V_{CORE}$ for Power Optimization

The concern posed by increasing MCLK frequency does not exist when decreasing MCLK, because the  $V_{CORE}$  level can still support MCLK frequencies below the ones for which they were intended. However, significant power efficiency gains can be achieved by operating  $V_{CORE}$  at the lowest value required for a given MCLK frequency. To decrease  $V_{CORE}$  level make sure the current MCLK frequency is supported at the new  $V_{CORE}$  level and then simply select the new power mode.

## 7.7 ARM Cortex Processor Sleep Modes

ARM Cortex™ processors have two instructions that can be used to place the processor in a sleep condition, namely WFI and WFE. In addition, another mode called sleep-on-exit is also available. These modes are also used by the PCM for entry into LPM0 and LPM3 and LPM4 modes of operation. These are described briefly in the following sections. Refer to the ARM documentation for complete details.

### 7.7.1 WFI, Wait For Interrupt

The WFI (wait for interrupt) instruction suspends code execution immediately. To wake from sleep following a WFI execution, the following conditions must occur:

- If the priority mask register is cleared (PRIMASK = 0) and an interrupt request occurs with its priority higher than the current base priority (BASEPRI), the interrupt causes the processor to wake from sleep and interrupt execution to occur. If the interrupt request priority is lower than the current base priority, the processor remains asleep and no interrupt execution takes place.
- If the priority mask register is set (PRIMASK = 1) and an interrupt request occurs with its priority higher than the current base priority (BASEPRI), the interrupt causes the processor to wake from sleep but the interrupt request is not serviced. If the interrupt request priority is lower than the current base priority, the processor remains asleep and also no interrupt execution takes place.

### 7.7.2 WFE, Wait For Event

The WFE (wait for event) instruction suspends code execution immediately if the event latch is cleared. If the event latch is set when a WFE execution occurs, the event latch is cleared and the processor continues with the next instruction. The wake from sleep following WFE execution depends also on the SEVONPEND bit in the SCR register. When SEVONPEND = 0, only enabled interrupts or events can wake the processor. Disabled interrupts are excluded. When SEVONPEND = 1, enabled events and all interrupts, including disabled interrupts, can wake the processor. To wake from sleep following a WFE execution, the following conditions must occur:

If SEVONPEND = 0:

- If the priority mask register is cleared (PRIMASK = 0) and an interrupt request occurs with its priority higher than the current base priority (BASEPRI), the interrupt causes the processor to wake from sleep and interrupt execution to occur. If the interrupt request priority is lower than the current base priority, the processor remains asleep and no interrupt execution takes place.
- If the priority mask register is set (PRIMASK = 1) and an interrupt request occurs with its priority higher than the current base priority (BASEPRI), the interrupt causes the processor to wake from sleep but the interrupt request is not serviced. If the interrupt request priority is lower than the current base priority, the processor remains asleep and also no interrupt execution takes place.

If SEVONPEND = 1:

- If the priority mask register is cleared (PRIMASK = 0), any interrupt (enabled or disabled) causes the processor to wake from sleep. Only interrupt requests higher than the current base priority (BASEPRI) cause interrupt execution to occur.
- If the priority mask register is set (PRIMASK = 1), any interrupt (enabled or disabled) causes the processor to wake from sleep. No interrupt execution takes place.

### 7.7.3 Sleep On Exit

ARM has a mode that allows the processor to automatically go back to sleep after an ISR completes. Setting SLEEPONEXIT = 1 in the System Control Register (SCR) causes the processor to go back to sleep when the interrupt service routine of the lowest pending priority interrupt completes. This allows the processor to be active only when an interrupt request is to be serviced.

### 7.7.4 SLEEPDEEP

The SCR contains a bit called SLEEPDEEP. This bit is used by ARM to differentiate a normal sleep ( $SLEEPDEEP = 0$ ) and a deep sleep ( $SLEEPDEEP = 1$ ). From the processor perspective either setting causes the processor to enter a sleeping state upon sleep entry. However, these settings can be used by the rest of the system to optimize power under different scenarios. Similarly, the PCM utilizes these bits to differentiate LPM0 and LPM3 and LPM4 modes. Hence when the application intends to put the device in LPM3 and LPM4 modes it must ensure that the SCR register gets programmed before executing WFI/WFE.

## 7.8 Power Mode Requests

The PCM Control 0 Register (PCMCTL0) is the primary mechanism for changing power modes. The PCMCTL0 contains two fields:

- Active Mode Request (AMR) : Used to change from one active mode to another.
- Low Power Mode Request (LPMR) : Used to enter LPM3, LPM4, LPM3.5, and LPM4.5 modes.

AMR has the higher priority over the LPMR settings. LPM3, LPM4, LPM3.5, and LPM4.5 modes are entered if the proper values are written into LPMR regardless of the AMR settings. In the event, there is a LPMR and AMR change that happen together along with SLEEPDEEP, the PCM first performs the AMR change and then enters into LPM3, LPM4, LPM3.5, or LPM4.5

The PCMCTL0 register is key protected for write access. This protects against inadvertent writes from changing the power mode settings. Read access of the PCMCTL0 is always possible and does not require a key.

The Current Power Mode (CPM) bits that reside in the PCMCTL0 register, are read only and reflect the current power mode of the system. CPM is updated when a power mode request completes.

## 7.9 Power Mode Selection

**Table 7-2** summarizes the various power modes available, the appropriate PCMCTL0 settings, and the entry mechanism for each of the modes.

**Table 7-2. Power Mode Selection<sup>(1)</sup>**

Mode	Description	PCMCTL0		SLEEPDEEP	Entry Mechanism
		AMR[3:0]	LPMR[3:0]		
AM_LDO_VCORE0	Active mode Core voltage level 0 LDO operation	0h	X	0	Writing of AMR register
AM_LDO_VCORE1	Active mode Core voltage level 1 LDO operation	1h	X	0	
AM_DCDC_VCORE0	Active mode Core voltage level 0 DC-DC operation	4h	X	0	
AM_DCDC_VCORE1	Active mode Core voltage level 1 DC-DC operation	5h	X	0	
AM_LF_VCORE0	Low-Frequency Active mode Core voltage level 0 LDO operation	8h	X	0	
AM_LF_VCORE1	Low-Frequency Active mode Core voltage level 1 LDO operation	9h	X	0	

<sup>(1)</sup> X = Do not care.

**Table 7-2. Power Mode Selection<sup>(1)</sup> (continued)**

Mode	Description	PCMCTL0		SLEEPDEEP	Entry Mechanism
		AMR[3:0]	LPMR[3:0]		
LPM0_LDO_VCORE0					
LPM0_DCDC_VCORE0					
LPM0_LF_VCORE0	LPM0 modes Core voltage level same as respective active mode	Same as the corresponding active mode. Programing AMR is not a pre-requisite for the device to enter LPM0.	X	0	WFI, WFE, Sleep-on-exit
LPM0_LDO_VCORE1					
LPM0_DCDC_VCORE1					
LPM0_LF_VCORE1					
LPM3	LPM3, LPM4 modes Core voltage level same as respective active mode	X	0h	1	WFI, WFE, Sleep-on-exit
LPM4					
LPM3.5	LPM3.5 mode Core voltage level 0	X	Ah	1	WFI, WFE, Sleep-on-exit
LPM4.5	LPM4.5 mode Core voltage turned off	X	Ch	1	WFI, WFE, Sleep-on-exit

## 7.10 Power Mode Transition Checks

PCM automatically checks for non-supported power mode transitions.

For active modes, any active mode transition not supported causes the active mode transition invalid flag to be set (AM\_INVALID\_TR\_IFG = 1) and the original AMR settings are retained. When AM\_INVALID\_TR\_IFG is set, this flag can be cleared by software by setting CLR\_AM\_INVALID\_TR\_IFG = 1. Setting AM\_INVALID\_TR\_IE = 1 enables an NMI/interrupt to be executed when AM\_INVALID\_TR\_IFG = 1. This is summarized in [Table 7-3](#).

**Table 7-3. AM Invalid Transition NMI/interrupt Enable**

AM_INVALID_TR_IE	AM_INVALID_TR_IFG	Response
0	0 or 1	none
1	0	none
	1	NMI/interrupt

For LPM3 and LPM4 modes, any active to LPM3 or LPM4 mode transition not supported cause the low-power mode transition invalid flag to be set (LPM\_INVALID\_TR\_IFG = 1) and the original AMR settings are retained. When LPM\_INVALID\_TR\_IFG is set, this flag can be cleared by software by setting CLR\_LPM\_INVALID\_TR\_IFG = 1. Setting LPM\_INVALID\_TR\_IE = 1 enables a NMI/interrupt to be executed when LPM\_INVALID\_TR\_IFG = 1. This is summarized in [Table 7-4](#).

**Table 7-4. LPM Invalid Transition NMI/Interrupt Enable**

LPM_INVALID_TR_IE	LPM_INVALID_TR_IFG	Response
0	0 or 1	none
1	0	none
	1	NMI/interrupt

## 7.11 Power Mode Clock Checks

PCM does not do any clock frequency/condition checks during any active/LPM0 power mode transitions. The application needs to ensure that the system conditions are met before the transition between these mode transitions.

For transitions to LPM3, LPM4, and LPMx.5 modes, the PCM does perform clock checking. When entering LPM3, LPM4, and LPMx.5 modes, it is possible that an active clock source is present in the system in any of the system clocks. This is typically defined as a static clock request. In cases like this, the application can decide whether to force the LPM3, LPM4, and LPMx.5 entry or not. Forcing LPM3, LPM4, and LPMx.5 entry in this condition is achieved by setting FORCE\_LPM\_ENTRY = 1. For example, the device is currently operating in AM\_LDO\_VCORE0 and has an active clock request on any of the clock lines (ACLK, MCLK, SMCLK, or HSMCLK). If a request is made to enter any of LPM3, LPM4, or LPMx.5 modes, the PCM denies the request and maintains its setting at AM\_LDO\_VCORE0 if FORCE\_LPM\_ENTRY = 0. This ensures that the active clock can be serviced reliably. The low-power mode invalid clock flag, LPM\_INVALID\_CLK\_IFG is set. When set, this flag remains set until cleared by software by setting CLR\_LPM\_INVALID\_CLK\_IFG = 1. Setting LPM\_INVALID\_CLK\_IE = 1 enables a NMI/interrupt to be executed when LPM\_INVALID\_CLK\_IFG = 1. This is summarized in [Table 7-5](#). The application must determine the proper action to be taken.

**Table 7-5. LPM Clock Checks NMI/Interrupt Enable**

LPM_INVALID_CLK_IE	LPM_INVALID_CLK_IFG	Response
0	0 or 1	none
1	0	none
	1	NMI/interrupt

---

**NOTE:** Application must ensure that FORCE\_LPM\_ENTRY bit is configured prior to LPMR bits while entering low-power modes. The device will go into indeterministic state if these two settings are programmed in wrong order while entering low-power modes.

---

## 7.12 Clock Configuration Changes

During any active mode, it is possible that the application may change the current clock configuration. It is the application's responsibility to ensure that any clock constraints for the current active mode are not violated. For example, when execution is in AM\_LF\_VCOREx, the maximum frequency supported for all system clocks is 128 kHz. This value must not be exceeded. The PCM does not adjust the power supply system for such clock violations.

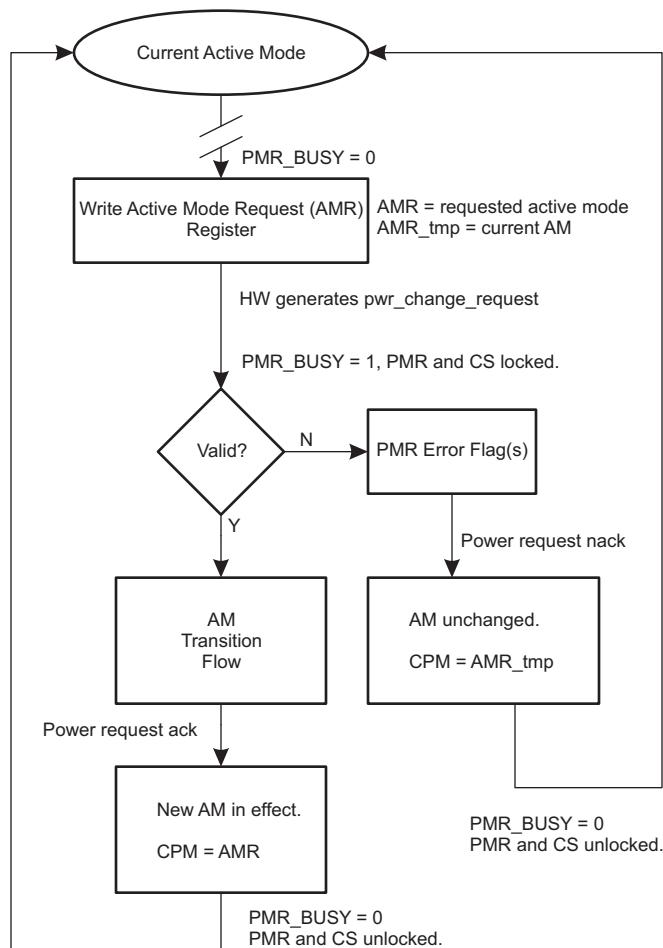
## 7.13 Changing Active Modes

Each active mode has a unique AMR setting as show in [Table 7-2](#). Writing to the AMR register causes an immediate active mode transition to begin. The application needs to understand the constraints on the system when switching from the current active mode to the requested active mode. In general, there are minimum supply, maximum frequency requirements and required flash wait state settings for the corresponding frequency of operation for each active mode of operation. The basic procedure is as follows:

1. The application selects the desired active mode by writing to AMR field of the PCMCTL0 using the correct PCMKEY. Any incorrect key causes the write to be ignored and the AMR contents remain at the original settings.
2. The writing of the PCMCTL0 causes a power change request to the PCM. The PCMCTL0 and the clock configuration registers of the CS are locked from any further write accesses. The CS also ignores any new clock requests that need a new oscillator source to be turned on. The PMR\_BUSY flag is set. Any writes to the PCMCTL0 or CS locked registers are ignored while PMR\_BUSY = 1.
3. The PCM validates the request. It checks if the active mode request is a valid transition. The active mode transition invalid flag, AM\_INVALID\_TR\_IFG, is set when the transition is not supported (see [Figure 7-3](#)). If the requested active mode has been invalidated, the AMR contents remain at the original settings and the requested active mode is not entered. If the AM\_INVALID\_TR\_IFG flag is set, it remains set until cleared by software. In this case, the PCM bypasses all subsequent steps and continues to the last step.
4. PCM processes any validated active mode request. The PCM configures the power supply system (PSS) appropriately for the requested active mode.

5. When the PCM completes its operations, PMR\_BUSY = 0 and the PCMCTL0 and CS registers are unlocked.

Figure 7-7 shows the basic flow as described above.



**Figure 7-7. Active mode transition flow.**

#### 7.13.1 DC-DC Error Checking

When using the DC-DC regulator, there may be situations where the regulator cannot remain active. For example, when the supply voltage drops below the minimum supply voltage necessary for the DC-DC operation. In this case, the DCDC\_ERROR\_IFG flag is set. When set, the DCDC\_ERROR\_IFG flag remains set until cleared by software by setting CLR\_DCDC\_ERROR\_IFG = 1. Setting DCDC\_ERROR\_IE = 1 enables an NMI/interrupt to be executed when DCDC\_ERROR\_IFG = 1 (see Table 7-6).

**Table 7-6. DC-DC Error NMI/interrupt Enable**

DCDC_ERROR_IE	DCDC_ERROR_IFG	Response
0	0 or 1	none
1	0	none
	1	NMI/interrupt

Any time the DCDC\_ERROR\_IFG flag is set, the PSS automatically switches to LDO as a backup regulator. The PSS also keeps trying to switch back to DC-DC operation by continuously monitoring the supply voltage. When the supply voltage is sufficient for DC-DC regulator, the PSS switches back to DC-DC mode of operation.

DCDC\_ERROR\_IFG is a sticky bit and is active until the error on DC-DC has been resolved. The application must ensure minimum supply for DC-DC regulator before it tries to switch over to the DC-DC mode of operation. Refer to the device-specific data sheet for the minimum supply required for the DC-DC to be operational in the device.

The DCDC\_ERROR\_IFG flag is triggered under the following scenarios:

1. DCDC\_ERROR\_IFG during an AM\_LDO\_VCOREx to AM\_DCDC\_VCOREx transition due to inadequate supply for DC-DC operation. In this case, the operating mode remains unchanged at AM\_LDO\_VCOREx.
2. DCDC\_ERROR\_IFG during steady state AM\_DCDC\_VCOREx operation due to insufficient supply for DC-DC operation. In this scenario, the application is notified about the DC-DC to LDO fail-safe operation. The application may then choose to change the power mode permanently back to LDO if the supply conditions for DC-DC cannot be met.

## 7.14 Entering LPM0 Modes

To enter LPM0 mode, it is required to utilize the processor's System Control Register (SCR). LPM0 mode entry is different to active mode requests, because the writing of the SCR register does not cause an immediate LPM0 mode entry. Only a WFI, WFE, or SLEEPONEXIT initiates the low-power mode entry process to begin. Using the SCR register can be considered as a way to precondition the device in preparation for LPM0 mode. For the majority of applications, the setting of these registers is rather infrequent and in some cases, only one time setup is necessary. The basic procedure is as follows:

1. The application selects sleep mode by writing SLEEPDEEP = 0 in the SCR.
2. At this point, the LPM0 mode is primed and waiting for a WFE, WFI, or sleep-on-exit event.

## 7.15 Exiting LPM0 Modes

The following events cause a wake up from LPM0 modes:

- External reset or NMI (RSTn/NMI)
- Enabled interrupt and latch events including general-purpose I/O events (refer to [Section 7.7](#) for details on wake-up criteria).
- Debugger events like halt, reset etc.

## 7.16 Entering LPM3 or LPM4 Modes

To enter LPM3 or LPM4 modes also, it is required to utilize the processor's System Control Register (SCR). LPM3 or LPM4 mode entry is different to active mode requests, because the writing of the SCR register does not cause an immediate LPM3 or LPM4 mode entry. Only a WFI, WFE, or SLEEPONEXIT initiates the low-power mode entry process to begin. Using the SCR register can be considered as a way to precondition the device in preparation for LPM3 or LPM4 modes. For the majority of applications, the setting of these registers is rather infrequent and in some cases, only one time setup is necessary.

The basic steps follow. The first three steps are required to prepare the device for LPM3 or LPM4 entry.

1. The application selects LPM3 or LPM4 mode by writing SLEEPDEEP = 1 in the SCR and LPMR = 0h in PCMCTL0.
2. At this time, LPM3 or LPM4 is waits for a WFE, WFI, or sleep-on-exit event.
3. On a WFI event, the PCM then checks whether this transition from the current active mode to LPM3 or LPM4 is valid. See [Figure 7-5](#) for the details on the valid LPM3 or LPM4 transitions. If this is not a valid transition, then PCM sets the LPM\_INVALID\_TR\_IFG flag and then aborts the LPM3 or LPM4 entry sequence. If, the LPM\_INVALID\_TR\_IFG = 1, and LPM\_INVALID\_TR\_IE = 0, the system enters a LPM0 mode corresponding to the active mode, till an interrupt event wakes it up.
4. If this is a valid transition, PCM locks the PCMCTL0 and CS registers. In addition, it disables any new

- clock requests and any new user POR [pin-based reset (RSTn/NMI) and debugger reset request (DBGRSTREQ)] requests from entering the system.
5. The PCM then checks the current clock settings to determine if there are any outstanding clock requests for clocks not supported during LPM3 mode. The clock invalid flag, LPM\_INVALID\_CLK\_IFG, is set when there are outstanding clock requests for clocks that are in violation of the LPM3 mode and FORCE\_LPM\_ENTRY = 0. The PCM does not enter the LPM3 mode when this condition occurs. If the clock settings are not in violation of the system clocks or when FORCE\_LPM\_ENTRY = 1, then the PCM starts to initialize the transition to LPM3. In this case, LPM\_INVALID\_CLK\_IFG is not set. In the event of LPM\_INVALID\_CLK\_IFG being set to 1, and LPM\_INVALID\_CLK\_IE = 0, the system enters the LPM0 mode corresponding to the active mode, until an interrupt event wakes it up.
  6. LPM4 mode is entered by programming for LPM3 mode with disabled RTC and WDT modules. LPM4 mode cannot be entered when the clock requirements for LPM3 mode are not met.
  7. The PCM then adjusts the power supply system for the LPM3 or LPM4 modes. When the power supply system has been adjusted and is stable, the PCM unlocks the PCMCTL0 and CS registers and enables clock requests. When the PCM completes its operations, the PMR busy flag is cleared (PMR\_BUSY = 0), and the PCMCTL0 and CS registers are unlocked.

---

**NOTE:** Prior to LPM3 entry, the application must ensure to configure the watchdog timer in the interval timer mode and not in the watchdog mode. Failure to do so might trigger a soft/hard reset condition during the low-power mode transitions. This might result in the system being in an indeterministic state.

---



---

**NOTE:** Clocks brought out on device pins also cause clock requests to the respective clocks in the device. The application should therefore treat these similar to other peripherals when entering LPM3 or LPM4 modes.

---



---

**NOTE:** Application must ensure that the analog modules (ADC14 or COMP\_E) must be disabled before entering LPM3 or LPM4 modes if they are configured to operate out of internal reference. This must be done regardless of the FORCE\_LPM\_ENTRY bit setting.

---

## 7.17 Exiting LPM3 or LPM4 Modes

There are multiple sources of wake up from LPM3/LPM4 modes. See [Table 7-8](#) for all possible wake-up sources from LPM3 or LPM4 modes. After wakeup, the device returns to the active mode from which LPM3 or LPM4 entry was initiated.

## 7.18 Entering LPM3.5 and LPM4.5 Modes

LPM3.5 and LPM4.5 entry and exit is handled differently than the other low-power modes. LPM3.5 and LPM4.5 modes, when used properly, gives the lowest power consumption available on a device. To achieve this, entry to LPM3.5 and LPM4.5 disables the majority of circuitry on the device by removing supply power. Because, removing the supply voltage from the circuitry, most register contents, as well as, SRAM contents are lost. For LPM3.5, the only modules available are the RTC and WDT, as well as, bank 0 of SRAM in retention mode. These modules are optionally active and are selectable by the user. For LPM4.5, no module functionality is available. In LPM4.5, the complete core logic supply is turned off and power is completely removed from all the circuitry. During LPM4.5 operation, only the minimum circuitry is powered that is required to wake-up the device.

The following are the basic steps for entry into LPM3.5 and LPM4.5 modes. Details about the operation of RTC and WDT, as well as, Digital I/O during LPMx.5 modes can be found in the respective module chapters.

- Ensure LOCKLPM5 = 0 and LOCKBKUP = 0, which may have been set from a previous LPMx.5 entry.
- Configure I/O appropriately. See the Digital I/O chapter for complete details on configuring I/O for LPMx.5 modes.
  - Set all ports to general purpose I/O. Configure each port to ensure no floating inputs based on the

#### application requirements

- If wake-up from I/O is desired, configure the wake-up I/O ports appropriately.
- For LPM3.5, if RTC operation is desired, enable the RTC. In addition, configure any RTC interrupts, if desired for a LPM3.5 wake-up event. See the RTC chapter for complete details.
- Enter LPM3.5 or LPM4.5 by writing LPMR = Ah or Ch, respectively and setting SLEEPDEEP = 1 in SCR.
- Device waits for WFI/WFE or sleep on exit event. Upon entering LPMx.5 mode, the I/O pin conditions remain locked at their current state. Also the LOCKLPM5, LOCKBKUP bits are automatically set in case of entry into LPM3.5 mode and only LOCKLPM5 bit is set in case of entry into LPM4.5 mode.

---

**NOTE:** Before LPM3.5 entry, the application must configure the watchdog timer in the interval timer mode and not in the watchdog mode. Failure to do so might trigger a soft or hard reset condition during the low-power mode transitions. This could result in the system being in an indeterministic state.

---

**NOTE:** Clocks brought out on device pins also cause clock requests to the respective clocks in the device. The application should therefore treat these similar to other peripherals when entering LPM3.5 or LPM4.5 modes.

---

## 7.19 Exiting LPM3.5 and LPM4.5 Modes

Exiting from LPM3.5 and LPM4.5 modes causes a POR event, which forces a complete reset of the system. Therefore, it is the application's responsibility to properly reconfigure the device upon exit from LPM3.5 and LPM4.5 modes. The wake-up times from LPM3.5 and LPM4.5 modes are significantly longer than the wake-up time from the other low-power modes (see the device-specific data sheet). Therefore, the usage of LPM3.5 and LPM4.5 modes should be restricted to very low duty cycle events. Any enabled wake-up event causes an exit from LPM3.5 and LPM4.5 modes.

Table 7-8 shows the different wake up sources supported for the LPM3.5 and LPM4.5 exit.

The following are the basic steps for exit from LPM3.5 and LPM4.5 modes. Details about the operation of RTC, WDT and Digital I/O during LPMx.5 modes can be found in the respective module chapters.

- LPM3.5 and LPM4.5 wake-up events (for example, I/O wake-up interrupt or RTC event) cause a POR event in the system which reinitializes the complete system. All peripheral registers are set to their default conditions.
- The PCMCTL0 register is cleared.
- The I/Os configured at entry into LPM3.5 and LPM4.5 modes retain their pin conditions due to LOCKLPM5 = 1. Keeping the I/O pins locked ensures that all pin conditions remain stable upon entering the active mode regardless of the default I/O register settings. All other port configuration register settings such as PxDIR, PxREN, PxOUT, PxDS, PxIES, and PxIE contents are lost.
- When in active mode, the I/O configuration and I/O interrupt configuration that was not retained during LPM3.5 and LPM4.5 modes should be restored to the values prior to entering LPM3.5 and LPM4.5 modes.
- If LPM3.5 was entered, the RTC interrupt configuration that was not retained in LPM3.5, should also be restored to the values prior to entering LPM3.5.
- At this point, the LOCKLPM5 and LOCKBKUP bits can be cleared. This releases the I/O pin conditions, as well as, the port and RTC interrupt configurations.
- NVIC interrupt enable register should be configured for port or RTC module if interrupt servicing is desired.
- To re-enter LPM3.5 and LPM4.5 modes, the LOCKLPM5 and LOCKBKUP bits must be cleared prior to re-entry, and the entry sequence to LPM3.5 and LPM4.5 should be followed.

---

**NOTE:** Any enabled wake-up event causes the device to exit LPM3.5 or LPM4.5 mode, regardless of priority. This differs from wake-up events during LPM0, LPM3, and LPM4 modes of operation.

---

## 7.20 Supply Voltage Supervisor and Monitor and Power Modes

The high-side supply voltage supervisor and monitor (SVSMH) can be enabled as required by the application. The SVSMH supports a full-performance mode and a low-power normal-performance mode. The full-performance mode has faster response times at the cost of additional power. The SVSMHLP bit in the PSS module selects the performance mode of SVSMH. Refer to the Power Supply System (PSS) chapter for details.

Specific power modes may require the full performance mode of SVSMH to ensure proper operation. The PCM ensures that this is adhered to based on the power mode request. In these cases, the PCM overrides the SVSMHLP setting. Note that the bit itself is not modified, only its effects on the system. Table 7-7 summarizes the SVSMH options with respect to each power mode.

**Table 7-7. SVSMH Performance and Power Modes**

Mode	PSS Bandgap	SVSMH	SVSMH mode when enabled
AM_LDO_VCORE0	Static	optional	full performance <sup>(1)</sup>
AM_LDO_VCORE1	Static	optional	full performance <sup>(1)</sup>
AM_DCDC_VCORE0	Static	optional	full performance <sup>(1)</sup>
AM_DCDC_VCORE1	Static	optional	full performance <sup>(1)</sup>
AM_LF_VCORE0	Static	optional	full performance <sup>(1)</sup>
AM_LF_VCORE1	Static	optional	full performance <sup>(1)</sup>
LPM0_LDO_VCORE0	Static	optional	full performance <sup>(1)</sup>
LPM0_LDO_VCORE1	Static	optional	full performance <sup>(1)</sup>
LPM0_DCDC_VCORE0	Static	optional	full performance <sup>(1)</sup>
LPM0_DCDC_VCORE1	Static	optional	full performance <sup>(1)</sup>
LPM0_LF_VCORE0	Static	optional	full performance <sup>(1)</sup>
LPM0_LF_VCORE1	Static	optional	full performance <sup>(1)</sup>
LPM3	Sampled	optional	selectable
LPM4	Sampled	optional	selectable
LPM3.5	Sampled	optional	selectable
LPM4.5	Sampled	optional	selectable

<sup>(1)</sup> Full performance mode is forced automatically by the PCM. SVSMHLP bit is do not care.

LPM4.5 mode can be entered with SVSMH enabled or disabled. Disabling the SVSMH results in lower power consumption, whereas enabling it provides the ability to detect supply drops and getting a "wake-up" due to the supply drop below the SVSMH threshold. Note that the wake-up due to a supply failure during LPM4.5 mode would be flagged as a SVSMH reset event. Enabling the SVSMH in LPM4.5 mode results additionally in a faster start-up time than with disabled SVSMH.

---

**NOTE:** Even if the SVSMH is configured as a monitor before entering LPM3.5 and LPM4.5 modes, it behaves as a supervisor while in these modes, and if triggered, issues a POR reset and not an interrupt.

---

## 7.21 Low-Power Reset

In battery operated applications it might be desirable to limit the current drawn by the device to a minimum after the supply dropped below the SVSMH power-down level. In such case the user should configure the SVSMH in the monitor mode and in the interrupt service routine the application can configure the I/Os into a defined state, disable any wake-up event (e.g from I/Os by clearing all PxIE bits), disable any module that can be operational in LPM3.5 mode like RTC or WDT, disable SVSMH by setting SVSMHOFF = 1, and then enter LPM4.5 mode.

In this low-power reset mode the SVSMH is disabled thus a complete power cycle (i.e. the supply needs to drop below the VCCDET power-down level) is required to reset and restart the device. Pulling the external reset pin (RSTn) low during the low-power reset state causes the device to enter its default reset state (with higher current consumption) and the device starts up when the supply rises above the SVSMH power-up level.

## 7.22 Power Requests During Debug

During a debug session, different requirements are necessary in order to support all the debug features of the system. For example, in LPM0 modes the processor and interface clocks are normally disabled. However, in a debug session, these clocks must remain active. This in turn leads to a different power setting requirement to serve the additional current required. Power measurements during a debug session may lead to different results compared to those during normal operation. The settings in the PCMCTL0 register reflect what the application code sees in normal operation.

### 7.22.1 Debug During Active Modes

The debugger can connect and halt the device anytime after the boot except the following conditions of device security:

- JTAG and SWD locked.
- IP protection active, with CPU executing in one of the secure memory zones.

Assuming that the device is in a non-secure mode, the user can plug in the debugger cable and issue a halt, thereby halting the CPU. Debugger can then run or step through the code or access the memory map.

When the user requests a connect:

- The debugger activates DBGPWRUPREQ and SYSPWRUPREQ through SWJ-DP. These are VCC domain signals.

If the user does a disconnect through the debugger command (controlled disconnect), the debugger issues a 'run free' command. At this point, the debugger cable can be removed safely. If CPU now runs into WFI or WFE, it stays in SLEEP because CPU is kept running and the debugger is disconnected. If the user disconnects the cable at any time (not recommended), the processor stays in whatever state it was before the cable was disconnected.

### 7.22.2 Debug During LPM0 Modes

The behavior is identical to debug when CPU is active. If the device is in LPM0 mode, and the debugger connects to the processor, the CPU is halted and the device enters the active mode the LPM0 was entered from. This discontinues the sleep mode of the CPU. The PC points to either the WFI or WFE instruction or, for the case of sleep-on-exit, at the first instruction after the return of the last pending interrupt. Stepping or running through simply re-execute the same instruction. A step is followed by a HALT, which also discontinues the SLEEP mode of the CPU.

If the user does a disconnect through the debugger command (controlled disconnect), the debugger issues a 'run free' command. At this point, the debugger cable can be removed safely. If CPU now runs into WFI or WFE, it stays in SLEEP because CPU is kept running, and the debugger is disconnected. If the user removes the cable at any time (not recommended), the processor is left in whatever state it was before the cable was disconnected.

### 7.22.3 Debug During LPM3, LPM4, and LPMx.5 Modes

Debug under LPM3, LPM4, and LPMx.5 modes behaves differently from the nondebug mode. If an active debug session is ongoing, the device does not enter LPM3, LPM4, or LPMx.5 mode but remains in LPM0 mode corresponding to the active mode. If, the device is already in the LPM3, LPM4, or LPMx.5 mode of operation and debug is initiated, then PCM initiates a wake up to the system to give control back to the debugger.

## 7.23 Wake-up Sources From Low Power Modes

[Table 7-8](#) lists all of the available wake-up sources from low-power modes on MSP432P4xx devices.

**Table 7-8. Wake-up Sources from Low Power Modes**

Peripheral	Wake-up Source	LPM0	LPM3	LPM4	LPM3.5	LPM4.5
eUSCI_A	Any enabled interrupt	Yes	–	–	–	–
eUSCI_B	Any enabled interrupt	Yes	–	–	–	–
Timer_A	Any enabled interrupt	Yes	–	–	–	–
Timer32	Any enabled interrupt	Yes	–	–	–	–
Comparator_E	Any enabled interrupt	Yes	–	–	–	–
ADC14	Any enabled interrupt	Yes	–	–	–	–
AES256	Any enabled interrupt	Yes	–	–	–	–
DMA	Any enabled interrupt	Yes	–	–	–	–
Clock System (CS)	Any enabled interrupt	Yes	–	–	–	–
Power Control Manager (PCM)	Any enabled interrupt	Yes	–	–	–	–
FLCTL	Any enabled interrupt	Yes	–	–	–	–
WDT_A in Watchdog Mode <sup>(1)</sup>	Watchdog driven reset	Yes	–	–	–	–
RTC_C	Any enabled interrupt <sup>(2)</sup>	Yes	Yes	–	Yes	–
WDT_A in Interval Timer Mode	Enabled interrupt	Yes	Yes	–	Yes	–
I/O Ports	Any enabled interrupt	Yes	Yes	Yes	Yes	Yes
NMI at Device Pin <sup>(3)</sup>	External NMI event	Yes	Yes	Yes	–	–
SVSMH in Monitor Mode (PSS) <sup>(4)</sup>	Enabled interrupt	Yes	Yes	Yes	–	–
Debugger Power up Request	SYSPWRUPREQ event	–	Yes	Yes	Yes	Yes
Debugger Reset Request	DBG_RSTREQ event <sup>(5)(6)</sup>	Yes	Yes	Yes	Yes	Yes
RSTn at Device Pin	External reset event <sup>(5)(6)</sup>	Yes	Yes	Yes	Yes	Yes
SVSMH in Supervisor Mode (PSS)	SVSMH driven reset <sup>(5)(6)</sup>	Yes	Yes	Yes	Yes	Yes
Power Cycle	Power on/off <sup>(5)(6)</sup>	Yes	Yes	Yes	Yes	Yes

<sup>(1)</sup> WDT\_A must be operated in interval timer mode during LPM3 and LPM3.5 modes for deterministic device operation.

<sup>(2)</sup> Refer to RTC\_C chapter for specific wake-up sources from low-power modes LPM3 and LPM3.5.

<sup>(3)</sup> RSTn/NMI device pin defaults to RSTn configuration during LPM3.5 and LPM4.5 modes.

<sup>(4)</sup> SVSMH defaults to supervisor mode during LPM3.5 and LPM4.5 modes.

<sup>(5)</sup> This event acts like a POR reset and takes the device to AM\_LDO\_VCORE0 configuration regardless of the mode from which the low-power mode was entered.

<sup>(6)</sup> LOCKLPM5 bit in case of LPM4.5 mode and both LOCKLPM5 and LOCKBKUP bits in case of LPM3.5 mode are cleared on wake-up due to the POR reset triggered by the wake-up event.

## 7.24 PCM Registers

The PCM module registers are listed in [Table 7-9](#). The base address can be found in the device-specific data sheet. The address offset is listed in [Table 7-9](#).

**Table 7-9. PCM Registers**

Offset	Acronym	Register Name	Type	Reset	Section
00h	PCMCTL0	Control 0 Register	Read/write	A5960000h	<a href="#">Section 7.24.1</a>
04h	PCMCTL1	Control 1 Register	Read/write	A5960000h	<a href="#">Section 7.24.2</a>
08h	PCMIE	Interrupt Enable register	Read/write	00000000h	<a href="#">Section 7.24.3</a>
0Ch	PCMIFG	Interrupt Flag Register	Read	00000000h	<a href="#">Section 7.24.4</a>
10h	PCMCLRIFG	Clear Interrupt Flag Register	Write	00000000h	<a href="#">Section 7.24.5</a>

---

**NOTE:** This is a 32-bit module and can be accessed through word (32-bit) or half-word (16-bit) or byte (8-bit) accesses.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 7.24.1 PCMCTL0 Register (offset = 00h) [reset = A5960000h]

PCM Control 0 Register

**Figure 7-8. PCMCTL0 Register**

31	30	29	28	27	26	25	24
PCMKEY							
rw-1	rw-0	rw-1	rw-0	rw-0	rw-1	rw-0	rw-1
23	22	21	20	19	18	17	16
PCMKEY							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0
15	14	13	12	11	10	9	8
Reserved		CPM					
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
LPMR				AMR			
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 7-10. PCMCTL0 Register Description**

Bit	Field	Type	Reset	Description
31-16	PCMKEY	RW	A596h	PCM key. Must write proper key to allow write access to PMR. Any incorrect key does not allow PMR to be written, and the power change request is not generated. Write PCMKEY = 695A_xxxxh to unlock for write access. Always reads back A596h. This is a word-accessible register. Any other type of access (half-word or byte access) is ignored by the PCM.
15-14	Reserved	R	0h	Reserved. Always read as 0.
13-8	CPM	R	0h	Current Power Mode. These bits reflect the current power mode and are updated when the power mode request completes. 0h = AM_LDO_VCORE0. LDO based Active Mode at Core voltage setting 0. 1h = AM_LDO_VCORE1. LDO based Active Mode at Core voltage setting 1. 2h = Reserved 3h = Reserved 4h = AM_DCDC_VCORE0. DC-DC based Active Mode at Core voltage setting 0. 5h = AM_DCDC_VCORE1. DC-DC based Active Mode at Core voltage setting 1. 6h = Reserved 7h = Reserved 8h = AM_LF_VCORE0. Low-Frequency Active Mode at Core voltage setting 0. 9h = AM_LF_VCORE1. Low-Frequency Active Mode at Core voltage setting 1. Ah-Fh = Reserved 10h = LPM0_LDO_VCORE0. LDO based LPM0 at Core voltage setting 0. 11h = LPM0_LDO_VCORE1. LDO based LPM0 at Core voltage setting 1. 12h = Reserved 13h = Reserved 14h = LPM0_DCDC_VCORE0. DC-DC based LPM0 at Core voltage setting 0. 15h = LPM0_DCDC_VCORE1. DC-DC based LPM0 at Core voltage setting 1. 16h = Reserved 17h = Reserved 18h = LPM0_LF_VCORE0. Low-frequency LPM0 at Core voltage setting 0. 19h = LPM0_LF_VCORE1. Low-frequency LPM0 at Core voltage setting 1. 1Ah-3Fh = Reserved.

**Table 7-10. PCMCTL0 Register Description (continued)**

Bit	Field	Type	Reset	Description
7-4	LPMR	RW	0h	<p>Low Power Mode Request. Used to request low power modes LPM3, LPM3.5, and LPM4.5. These bits can only be modified while PMR_BUSY = 0 of the PCMCTL1.</p> <p>0h = LPM3. Core voltage setting is similar to the mode from which LPM3 is entered.</p> <p>1h-9h = Reserved<sup>(1)</sup></p> <p>Ah = LPM3.5. Core voltage setting 0.</p> <p>Bh = Reserved.<sup>(1)</sup></p> <p>Ch = LPM4.5</p> <p>Dh-Fh = Reserved<sup>(1)</sup></p>
3-0	AMR	RW	0h	<p>Active Mode Request. Used to request active modes. These bits can only be modified while PMR_BUSY = 0 of the PCMCTL0.</p> <p>0h = AM_LDO_VCORE0. LDO based Active Mode at Core voltage setting 0.</p> <p>1h = AM_LDO_VCORE1. LDO based Active Mode at Core voltage setting 1.</p> <p>2h-3h = Reserved<sup>(2)</sup></p> <p>4h = AM_DCDC_VCORE0. DC-DC based Active Mode at Core voltage setting 0.</p> <p>5h = AM_DCDC_VCORE1. DC-DC based Active Mode at Core voltage setting 1.</p> <p>6h-7h = Reserved<sup>(2)</sup></p> <p>8h = AM_LF_VCORE0. Low-Frequency Active Mode at Core voltage setting 0.</p> <p>9h = AM_LF_VCORE1. Low-Frequency Active Mode at Core voltage setting 1.</p> <p>Ah-Fh = Reserved<sup>(2)</sup></p>

<sup>(1)</sup> Writing reserved values into LPMR can cause nondeterministic device behavior

<sup>(2)</sup> Writing reserved values into AMR can cause nondeterministic device behavior.

### 7.24.2 PCMCTL1 Register (offset = 04h) [reset = A5960000h]

PCM Control 1 Register

**Figure 7-9. PCMCTL1 Register**

31	30	29	28	27	26	25	24
PCMKEY							
rw-1	rw-0	rw-1	rw-0	rw-0	rw-1	rw-0	rw-1
23	22	21	20	19	18	17	16
PCMKEY							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0
15	14	13	12	11	10	9	8
Reserved						PMR_BUSY	
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved				FORCE_LPM_ENTRY		LOCKBKUP	LOCKLPM5
r-0	r-0	r-0	r-0	r-0	rw-0	rw-(0)	rw-(0)

**Table 7-11. PCMCTL1 Register Description**

Bit	Field	Type	Reset	Description
31-16	PCMKEY	RW	A596h	PCM key. Must write proper key to allow write access to PCMCTL1. Any incorrect key does not allow PCMCTL1 to be written, and the write is ignored. Write PCMKEY = 695A_xxxxh to unlock for write access. Always reads back A596h. This is a word-accessible register. Any other type of access (half-word or byte access) is ignored by the PCM.
15-9	Reserved	R	0h	Reserved. Reads back 0.
8	PMR_BUSY	R	0h	Power mode request busy flag. This flag is set while a power change request is being processed. The flag is cleared when the request has completed.  Writes to the PCMCTL0 or Clock System registers are ignored while PMR_BUSY = 1.  Reads to the PCMCTL0 or Clock System registers are possible while PMR_BUSY = 1.
7-3	Reserved	R	0h	Reserved. Reads back 0.
2	FORCE_LPM_ENTRY	RW	0h	Bit selection for the application to determine whether the entry into LPM3/LPMx.5 should be forced even if there are active system clocks running which do not meet the LPM3/LPMx.5 criteria. This bit by itself does not cause a LPM3/LPMx.5 by itself, but is a pre-requisite before the CPU executes WFI. This bit needs to be programmed prior to LPMR bits in PCMCTL0 register.  0b = PCM aborts LPM3 or LPMx.5 transition if the active clock configuration does not meet the LPM3 or LPMx.5 entry criteria. PCM generates the LPM_INVALID_CLK flag on abort to LPM3 or LPMx.5 entry.  1b = PCM enters LPM3 or LPMx.5 after shutting off the clocks forcefully. The application must make sure that the RTC and WDT are clocked using BCLK tree to keep these modules active in LPM3 in LPM3.5. In LPM4.5, all clocks are turned off and the core voltage is turned off.
1	LOCKBKUP	RW	0h	Lock Backup. After this bit is set, it can be cleared only by the application or by a power cycle. This bit is automatically set upon LPM3.5 entry. The application cannot set this bit.  In the write mode, this bit should always be written with a value of 0 to clear it. Writing a value of 1 has no effect on the system and these writes are ignored.  0b = Backup domain configuration defaults to reset condition. 1b = Backup domain configuration remains locked during LPM3.5 entry and exit.

**Table 7-11. PCMCTL1 Register Description (continued)**

Bit	Field	Type	Reset	Description
0	LOCKLPM5	RW	0h	<p>Lock LPM5. When this bit is set, it can be cleared only by the application or by a power cycle. This bit is automatically set upon LPM3.5 or LPM4.5 entry. The application cannot set this bit.</p> <p>In the write mode, this bit should always be written with a value of 0 to clear it. Writing a value of 1 has no effect on the system and these writes are ignored.</p> <p>0b = LPMx.5 configuration defaults to reset condition. 1b = LPMx.5 configuration remains locked during LPMx.5 entry and exit.</p>

### 7.24.3 PCMIE Register (offset = 08h) [reset = 00000000h]

PCM Interrupt Enable Register

**Figure 7-10. PCMIE Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved	DCDC_ERROR_IE	Reserved			AM_INVALID_TR_IE	LPM_INVALID_CLK_IE	LPM_INVALID_TR_IE
r-0	rw-0	r-0	r-0	r-0	rw-0	rw-0	rw-0

**Table 7-12. PCMIE Register Description**

Bit	Field	Type	Reset	Description
31-7	Reserved	R	0h	Reserved. Reads back 0.
6	DCDC_ERROR_IE	RW	0h	DC-DC error interrupt enable. Setting this bit enables an interrupt/NMI when DC-DC operation cannot be achieved or maintained. 0b = Disabled 1b = Enabled
5-3	Reserved	R	0h	Reserved. Reads back 0.
2	AM_INVALID_TR_IE	RW	0h	Active mode invalid transition interrupt enable. Setting this bit enables an interrupt/NMI on an invalid transition setting during an active power mode request. 0b = Disabled 1b = Enabled
1	LPM_INVALID_CLK_IE	RW	0h	LPM invalid clock interrupt enable. Setting this bit enables an interrupt/NMI on an invalid clock setting during a LPM3/LPMx.5 transition from an active mode when FORCE_LPM_ENTRY = 0. This bit has no effect when FORCE_LPM_ENTRY = 1. 0b = Disabled 1b = Enabled
0	LPM_INVALID_TR_IE	RW	0h	LPM invalid transition interrupt enable. Setting this bit enables an interrupt/NMI on an invalid transition from Active Mode to LPM3 (LPMx.5 all transitions are allowed). 0b = Disabled 1b = Enabled

**7.24.4 PCMFIFG Register (offset = 0Ch) [reset = 00000000h]**

PCM Interrupt Flag Register

**Figure 7-11. PCMFIFG Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
7	6	5	4	3	2	1	0
Reserved	DCDC_ERROR_IFG	Reserved			AM_INVALID_TR_IFG	LPM_INVALID_CLK_IFG	LPM_INVALID_TR_IFG
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

**Table 7-13. PCMFIFG Register Description**

Bit	Field	Type	Reset	Description
31-7	Reserved	R	0h	Reserved. Reads back 0.
6	DCDC_ERROR_IFG	R	0h	DC-DC error flag. This flag is set if DC-DC operation cannot be achieved or maintained. Flag remains set until cleared by software.
5-3	Reserved	R	0h	Reserved. Reads back 0.
2	AM_INVALID_TR_IFG	R	0h	Active mode invalid transition flag. This flag is set if the active mode request is an invalid transition. Flag remains set until cleared by software.
1	LPM_INVALID_CLK_I FG	R	0h	LPM invalid clock flag. This flag is set if the LPM request is invalid due to a clock request active before the LPM3/LPMx.5 entry while the FORCE_LPM_ENTRY = 0. Flag remains set until cleared by software.
0	LPM_INVALID_TR_IF G	R	0h	LPM invalid transition flag. This flag is set if the requested Active Mode to LPM3 transition is invalid. Flag remains set until cleared by software.

**7.24.5 PCMCLRIFG Register (offset = 10h) [reset = 00000000h]**

PCM Clear Interrupt Flag Register

**Figure 7-12. PCMCLRIFG Register**

31	30	29	28	27	26	25	24
Reserved							
w1	w1	w1	w1	w1	w1	w1	w1
23	22	21	20	19	18	17	16
Reserved							
w1	w1	w1	w1	w1	w1	w1	w1
15	14	13	12	11	10	9	8
Reserved							
w1	w1	w1	w1	w1	w1	w1	w1
7	6	5	4	3	2	1	0
Reserved	CLR_DCDC_E RROR_IFG	Reserved			CLR_AM_INVA LID_TR_IFG	CLR_LPM_INV ALID_CLK_IFG	CLR_LPM_INV ALID_TR_IFG
w1	w1	w1	w1	w1	w1	w1	w1

**Table 7-14. PCMCLRIFG Register Description**

Bit	Field	Type	Reset	Description
31-7	Reserved	W	0h	Reserved. Write 1 only. Reads back 0.
6	CLR_DCDC_ERROR_IFG	W	0h	Clear DC-DC error flag. 0b = No effect 1b = Clear flag
5-3	Reserved	W	0h	Reserved. Write 1 only. Reads back 0.
2	CLR_AM_INVALID_TR_IFG	W	0h	Clear active mode invalid transition flag. 0b = No effect 1b = Clear flag
1	CLR_LPM_INVALID_CLK_IFG	W	0h	Clear LPM invalid clock flag. 0b = No effect 1b = Clear flag
0	CLR_LPM_INVALID_TR_IFG	W	0h	Clear LPM invalid transition flag. 0b = No effect 1b = Clear flag

## ***Flash Controller (FLCTL)***

This chapter describes the Flash controller.

Topic	Page
<b>8.1 Introduction .....</b>	<b>377</b>
<b>8.2 Common Operations Using the Flash Controller .....</b>	<b>378</b>
<b>8.3 Advanced Operations using the Flash Controller .....</b>	<b>380</b>
<b>8.4 FLCTL Registers.....</b>	<b>395</b>

## 8.1 Introduction

The flash memory is byte, word (4 bytes), and full-word (16 bytes) addressable and programmable. The Flash controller acts as the control and access interface between software (application) and the various functions supported by the Flash memory on the device. The flash controller features include:

- Internal programming voltage generation
- Byte, word (4 bytes), full-word (16 bytes), and burst (up to 4x16 bytes) programmable
- Ultra-low-power operation
- Sector erase and mass erase
- Optimized read operations from the Flash (for program fetches or data reads)
- Configurable write and erase protection per sector

### 8.1.1 Flash Memory Organization

The Flash memory on the device consists of two independent equal sized memory banks, each containing the following regions

- Main Memory region: This is the primary code memory and is intended for code and data for the user application.
- Information Memory region: Intended for TI or customer code or data. Some of the Information memory sectors are used by TI, and others are available for users. Details on Information memory are available in the device-specific data sheet.

### 8.1.2 Flash Controller Address Mapping

The two memory regions implemented in the Flash are equally divided between the two Flash banks. The following example is for the MSP432P401R MCU, which has 256KB of flash main memory and 16KB of information memory.

- Main Memory, 256 KB, mapped from 0h through 3\_FFFFh
  - Accesses from 0h through 1\_FFFFh are mapped to Bank0. All Bank0 parameters and settings (through configuration registers) apply to these accesses.
  - Accesses from 2\_0000h through 3\_FFFFh are mapped to Bank1. All Bank1 parameters and settings (through configuration registers) apply to these accesses.
- Information Memory, 16 KB, mapped from 20\_0000h through 20\_3FFFh
  - Accesses from 20\_0000h through 20\_1FFFh are mapped to Bank0. All Bank0 parameters and settings (through configuration registers) apply to these accesses.
  - Accesses from 20\_2000h through 20\_3FFFh are mapped to Bank1. All Bank1 parameters and settings (through configuration registers) apply to these accesses.

Any access that does not fall within a valid Flash memory region returns a bus error response.

---

**NOTE:** The previous address mapping example is for illustration purposes ONLY. See the appropriate data sheet for the Flash region addresses on the device of interest.

---

### 8.1.3 Flash Controller Access Privileges

The Flash memory on the device can be accessed by the CPU, the DMA, or through the debugger (JTAG or SW):

- CPU (Instruction and Data buses)
  - The CPU can issue instruction fetches to the entire Flash memory region
  - The CPU can issue data reads and writes to the entire Flash memory region (unless the access is blocked by the device security architecture as described in the SYSCTL chapter).
- DMA
  - The DMA has conditional access read permissions to the Flash. If the device is not secure, or JTAG and SWD lock-based security is active, the DMA has full permissions to the entire Flash

space. If IP protection is active on the device, the DMA is allowed reads and writes to Bank1 only. In this case, DMA accesses to Bank0 returns a bus error response.

- Debugger
  - A debugger can initiate accesses to the Flash. If the device is non-secure, all debugger accesses are permitted. However, if the device is enabled for any form of code security, debugger accesses to the Flash is denied by the device security architecture.

---

**NOTE:** For details on how Flash access privileges are controlled by security, see [Section 4.8](#) in the *System Controller (SYSCTL)* chapter.

---

## 8.2 Common Operations Using the Flash Controller

### 8.2.1 Using MSP432 Driver Library for Flash Operations

The MSP432 Driver Library allow the Flash Module to be accessed through a simple and easy-to-use interface for a number of commonly used flash operations such as read, write and erase. TI recommends incorporating MSP432 Driver Library APIs for all flash operations to ensure safe execution of flash routines as per the required specification.

A full listing of MSP432 Driver Library APIs is available in the [MSP432 Driver Library MSP432P4xx User's Guide](#) (see Chapter 8, *Flash Memory Controller*).

### 8.2.2 Flash Read

Flash read operations involve a data value being output by the Flash Memory.

#### 8.2.2.1 Flash Read Timing Control and Wait States

The Flash Controller is configurable in terms of the number of memory bus cycles it takes to service any read command. This allows the CPU execution frequency to be higher than the maximum read frequency supported by the Flash Memory. If the bus clock speed is higher than the native frequency of the Flash, the access is stalled for the configured number of wait states, allowing data from the Flash to be accessed reliably.

User software is required to program the number of wait-states into these registers based on the CPU's execution frequency.

MSP432 Driver Library APIs in [Table 8-1](#) can be used to set the Flash wait state.

**Table 8-1. MSP432 Driver Library API for Flash Wait-State Configuration**

MSP432 Driver Library API	Function
FlashCtl_setWaitState	Changes the number of wait states that are used by the flash controller for read operations
FlashCtl_getWaitState	Returns the set number of flash wait states for the given flash bank

---

**NOTE:** Refer to the device data sheet for CPU execution frequency and wait-state requirements.

---

#### 8.2.2.2 Read Buffering

The MSP432P device flash is organized with a line size of 128 bits. To offer optimal power consumption and performance across predominantly contiguous memory accesses, the flash controller offers a "Read Buffering" feature. If Read Buffering is enabled, the flash memory always reads an entire 128-bit line irrespective of the access size of 8, 16, or 32 bits. The 128-bit data and its associated address is internally buffered by the flash controller, so subsequent accesses (expected to be contiguous in nature) within the same 128-bit address boundary are serviced by the buffer. Hence, the flash accesses see wait-states only

when the 128-bit boundary is crossed, while read accesses within the buffer's range are serviced without any bus stalls. If read buffering is disabled, accesses to the flash bypasses the buffer, and the data read from the flash is limited to the width of the access (8, 16, or 32 bits). Each bank has independent settings for the read buffering. In addition, within each bank, the application has independent flexibility to enable read buffering for instruction and data fetches.

Read buffers are bypassed during any program or erase operation by the controller to ensure data coherency.

Read buffers are disabled by default.

The MSP432 Driver Library APIs in [Table 8-2](#) can be used to enable and disable read buffering.

**Table 8-2. MSP432 Driver Library API for Flash Read Buffering Configuration**

MSP432 Driver Library API	Function
FlashCtl_enableReadBuffering	Enables read buffering on accesses to a specified bank of flash memory
FlashCtl_disableReadBuffering	Disable read buffering on accesses to a specified bank of flash memory

### 8.2.3 Flash Program

Programming of bits in the Flash memory involves setting the targeted bits to the value 0. The Flash memory architecture supports programming of bits from a single bit up to four entire flash word width (128-bits) in one program operation

MSP432 Driver Library API in [Table 8-3](#) can be used to program memory. This function can be used to program memory blocks of any size from a single byte to larger blocks. It is a blocking function that disables Master interrupts before the start of flash programming. For more information on how this function works refer to the [MSP432 Peripheral Driver Library User's Guide](#).

**Table 8-3. MSP432 Driver Library API for Flash Program Operation**

MSP432 Driver Library API	Function
FlashCtl_programMemory	Program a portion of flash memory with the provided data

### 8.2.4 Flash Erase

The logical value of an erased flash memory bit is 1. Each bit can be programmed from 1 to 0 individually, but to reprogram from 0 to 1 requires an erase cycle. The smallest amount of flash that can be erased in both Main and Information memory regions is one sector (4KB). There are two erase modes offered by the Flash memory: Sector Erase and Mass Erase.

#### 8.2.4.1 Sector Erase Mode

In the sector erase mode, the Flash Controller can be configured to erase a sector of the flash. This sector can be in either bank, in the information memory or the main memory region.

MSP432 Driver Library API in [Table 8-4](#) can be used to erase a flash sector. This function is blocking and does not exit until operation has either completed or failed due to an error. Master interrupts are disabled throughout execution of this function.

**Table 8-4. MSP432 Driver Library API for Flash Sector Erase Operation**

MSP432 Driver Library API	Function
FlashCtl_eraseSector	Erases a sector of MAIN or INFO flash memory.

#### 8.2.4.2 Mass Erase Mode

In the mass erase mode, the Flash Controller is set up to erase the entire Flash Memory. Mass erase is simultaneously applied to both Banks.

MSP432 Driver Library API in [Table 8-5](#) can be used to mass erase flash. This function is blocking and does not exit until operation has either completed or failed due to an error. Master interrupts are disabled throughout execution of this function.

**Table 8-5. MSP432 Driver Library API for Flash Mass Erase Operation**

MSP432 Driver Library API	Function
FlashCtl_performMassErase	Performs a mass erase on all unprotected flash sectors. Protected sectors are ignored.

#### 8.2.5 Flash Program and Erase Protection

The Flash Memory implements one PROT (write and erase protection) bit per sector of Flash memory. If the PROT bit is set to 1, that sector becomes a read-only type of memory, and any program or erase commands to addresses in that sector are not honored. The Flash Controller provides access to PROT configuration registers for the Main and Information memory regions, with one bit dedicated to each Flash sector. Using these bits, the application can protect sectors from inadvertent program or erase operations. In addition, these bits can also be used to optimize erase timing.

For example, if the application is required to erase a portion of the main memory, say 128kB out of 256kB. In such a case one option is to carry out the erase as 32 individual sector erase operations. Alternatively, the application can set the PROT bits of the targeted 128kB of memory to 0. All other PROT bits for the Main and Information memory space are set to 1. This is followed by initiating a mass erase operation, which clears the targeted 128kB in one erase cycle, thereby saving time and energy overhead.

---

**NOTE:** If IP protection is enabled, the configuration of the PROT bit for the IP protected sectors of flash memory is ignored by the Flash Controller.

---

MSP432 Driver Library API in [Table 8-6](#) can be used to setup program or erase protection.

**Table 8-6. MSP432 Driver Library API for Setting up Program or Erase Operation**

MSP432 Driver Library API	Function
FlashCtl_protectSector	Enables program protection on the given sector mask.
FlashCtl_unprotectSector	Disables program protection on the given sector mask.

### 8.3 Advanced Operations using the Flash Controller

The MSP432 Driver Library allow the Flash Module to be accessed through a simple and easy-to-use interface for a number of commonly used flash operations such as read, write and erase. TI recommends incorporating MSP432 Driver Library APIs for all flash operations to ensure safe execution of flash routines as per the required specification. A full listing of MSP432 Driver Library APIs is available in the [MSP432 Driver Library MSP432P4xx User's Guide](#) (see Chapter 8, *Flash Memory Controller*).

In applications where custom software is used for in-system program or erase of flash, it is important to know that programming and erasing MSP432 flash requires verification stages in addition to the write/erase operations. Failure to implement the required verification stages and to follow the exact routines listed in the "Software Flow" sections could result in the MSP432 Flash being programmed or erased incorrectly. TI recommends users read and understand the requirements for custom in-system programming before implementation.

#### 8.3.1 Advanced Flash Read

The MSP432P flash controller can be programmed to output data from the flash memory in different read modes. The different read modes supported on the device are:

- Normal Read: This is the most commonly used mode. Users must ensure that CPU execution happens only in this mode.
- Read Margin 0/1: Marginal read modes are primarily test modes to check flash memory. These modes are useful in determining the margins of programmed bits after a long duration of device operation in the field.
- Program Verify: This read mode helps check if the memory is programmed with sufficient margin. This mode is used whenever a user sees a post verify error during program operations.
- Erase Verify: This read mode helps check if the memory is erased with sufficient margin. This mode should be used following every erase operation.

The required read mode can be enabled using the RD\_MODE bits in the FLCTL\_BANKx\_RDCTL register.

It is important to note that each flash bank has its own individual settings for the read modes. All accesses to the bank are carried out as per the configured read mode.

---

**NOTE:** Due to mode transition latencies, there is a delay between configuration of the new mode setting for a Bank and the time it is actually ready to process reads in that mode. In addition, the Flash Controller may stall the transition of a bank into a particular mode because of a simultaneous operation on the other bank. After changing the mode setting, it is the application's responsibility to ensure that the read mode status for the bank reflects the new mode. This ensures that all subsequent reads happen in the targeted mode; otherwise, a few of the reads to the Bank may continue to be serviced in the old mode.

---

MSP432 Driver Library API in [Table 8-7](#) can be used to configure the read mode.

**Table 8-7. MSP432 Driver Library API for Setting up Flash Read Modes**

MSP432 Driver Library API	Function
FlashCtl_setReadMode	Sets the flash read mode to be used by flash read operations
FlashCtl_getReadMode	Gets the flash read mode to be used by flash read operations.

### 8.3.1.1 Burst Read and Compare Feature

The flash controller supports a Burst Read and Compare feature, which permits a fast read and compare operation on a contiguous section of the Flash memory. Implementing a burst read permits the flash controller to optimize the time taken for the operation by comparing all 128 bits at a time.

The burst read and compare feature is useful for the verification of the memory post an erase. This is done in the Erase Verify read mode of operation.

### 8.3.2 Advanced Flash Program

Programming MSP432 flash requires the following stages to be incorporated into user software:

1. Pre-program-verify
2. Initiate Program
3. Post-program-verify

The MSP432 Flash controller requires that user application implements the exact routines listed in [Section 8.3.2.2.1](#) or [Section 8.3.2.3.1](#) depending on the programming mode used. Failure to follow the documented software flow could result in the MSP432 Flash being programmed incorrectly. TI recommends users read and understand the requirements for custom in-system programming before implementation.

The pre-program-verify stage is not mandatory if the application knows that the flash location to be programmed is already in erased state. However, post-program-verify is required following every program operation.

The Flash Controller implements both pre-program-verify and post-program-verify stages in hardware using the auto-verify feature. The auto-verify feature is explained in detail in [Section 8.3.2.1](#). The flash programming stage can be accomplished using any one of the advanced program modes listed below:

- Immediate write mode
- Full-word programming mode
- Burst program mode.

Additionally, it should be noted that since the Flash controller is optimized to provide energy and time-efficient program operations, user application software may be required to execute the three stages of programming multiple times, thereby subjecting the flash memory to multiple "pulses". However, each device supports a maximum number of program pulses that is defined by the Flash Maximum Programming Pulses parameter in the device data sheet, Device Descriptors (TLV) section.

### 8.3.2.1 Using the Auto-Verify Feature

To prevent accidental over-programming of a bit or to check that a bit has been sufficiently programmed, the Flash Controller provides control bits to implement an automatic program-verify and compare operation before and after each programming cycle. These are called the automatic pre-program-verify and post-program-verify respectively.

Automatic preprogram-verify is not mandatory if the application knows that the flash location to be programmed is already in erased state. However, automatic post-program-verify is required following every program operation.

When pre-program-verify is enabled, flash controller initiates a read to the address to be programmed in program-verify read mode. The flash controller then compares the data received with the value to be programmed. It issues an error if any of the bits to be programmed already shows a state of 0 in the memory. This error is indicated by the AVPRE flag in the FLCTL\_IFG Register.

The Post-program-verify operation initiates a read to the address after programming is completed. This read is also initiated in the program-verify read mode. The flash controller then compares the data received with the value that was intended to be programmed and issues an error if any of the bits that were supposed to be programmed show a state of 1 in the memory. This error is indicated by the AVPST flag in the FLCTL\_IFG Register.

Depending on the programming mode used the auto-verify feature can be configured using the bits shown in [Table 8-8](#)

**Table 8-8. Configuring the Auto-Verify Mode Through Direct Register Access**

Programming Mode	Auto-Verify Function	Bit	Register
Immediate and Full word	Pre-program-verify	VER_PRE	FLCTL_PRG_CTLSTAT
Immediate and Full word	Post-program-verify	VER_PST	FLCTL_PRG_CTLSTAT
Burst	Pre-program-verify	AUTO_PRE	FLCTL_PRGBRST_CTL
Burst	Post-program-verify	AUTO_PST	FLCTL_PRGBRST_CTL

Alternatively, this feature can be configured using MSP432 Driver Library API shown in [Table 8-9](#)

**Table 8-9. MSP432 Driver Library API for Setting up Auto-Verify Before Program Operations**

MSP432 Driver Library API	Function
FlashCtl_setProgramVerification	Sets up pre- or post-verification of burst and regular flash programming instructions
FlashCtl_clearProgramVerification	Clears pre- or post-verification of burst and regular flash programming instructions.

### 8.3.2.2 Flash Programming Using Immediate and Full-Word Modes

When the immediate write mode of programming is configured, the flash controller initiates a program operation immediately upon receiving a write command.

To optimize write latencies and the power consumption during Flash program operations, the application can configure the Flash Controller to buffer multiple writes from the CPU and initiate the program operation only after a complete 128-bit flash word has been composed. This method of programming, enabled by the full-word programming mode, is useful in cases where the application is working with larger word sizes and prefers to initiate writes only when it has at least 16 bytes of data ready for programming.

The following steps explain how full-word programming mode should be used by the application:

1. The data must be written in an incrementing address fashion, starting with a 128-bit aligned LSB.
  - Can be written as 4x32-bit writes, starting from the least significant 32-bit word.
  - Can be written as 8x16-bit writes, starting from the least significant 16-bit word
  - Can be written as 16x8-bit writes, starting from the least significant byte
  - Can be written as a combination of the above, but must start with the LSB being loaded with the first write and end with the most significant byte being loaded with the last write
2. When the above is completed, the Flash Controller now has a complete 128-bit write word that is used for the program operation.

---

**NOTE:** The application can write a single byte to the LSB (to start the word composition), and a single byte to the MSB of the 128-bit address (to end the word composition). This also results in the Flash controller composing a full 128-bit word in which only the least and most significant bytes have actual data, while the intermediate data bits are all filled with 1s (masked from programming).

---



---

**NOTE:** It is the application's responsibility to ensure that the writes in this mode follow the correct sequence (LSB initiation and MSB completion) for the program operation to complete in a deterministic fashion. If the 128-bit boundary is crossed before the word is completely composed, the buffer starts composing a new word starting with the new 128-bit address provided to it. Previously stored data in the buffer is discarded to ensure that data is not written to an incorrect location altogether. Similarly, if the first write does not start at a 128-bit boundary, the buffer does not start composition and the data is discarded. The Flash Controller sets an interrupt flag in both of these cases (which can be enabled to generate an interrupt).

---

The full-word programming mode is enabled by setting the MODE bit in the FLCTL\_PRG\_CTLSTAT register.

Alternatively, this feature can be configured using the following MSP432 Driver Library API in [Table 8-10](#)

**Table 8-10. MSP432 Driver Library API for Enabling Program Operations**

MSP432 Driver Library API	Function
FlashCtl_enableWordProgramming	Enables word programming of flash memory.
FlashCtl_disableWordProgramming	Disables word programming of flash memory.
FlashCtl_isWordProgrammingEnabled	Returns if word programming mode is enabled (and if it is, the specific mode)

### 8.3.2.2.1 Software Flow for Immediate and Full-Word Programming Modes

Figure 8-1, Figure 8-2, and Figure 8-3 show the software flow to be followed for reliable immediate and full word programming modes.

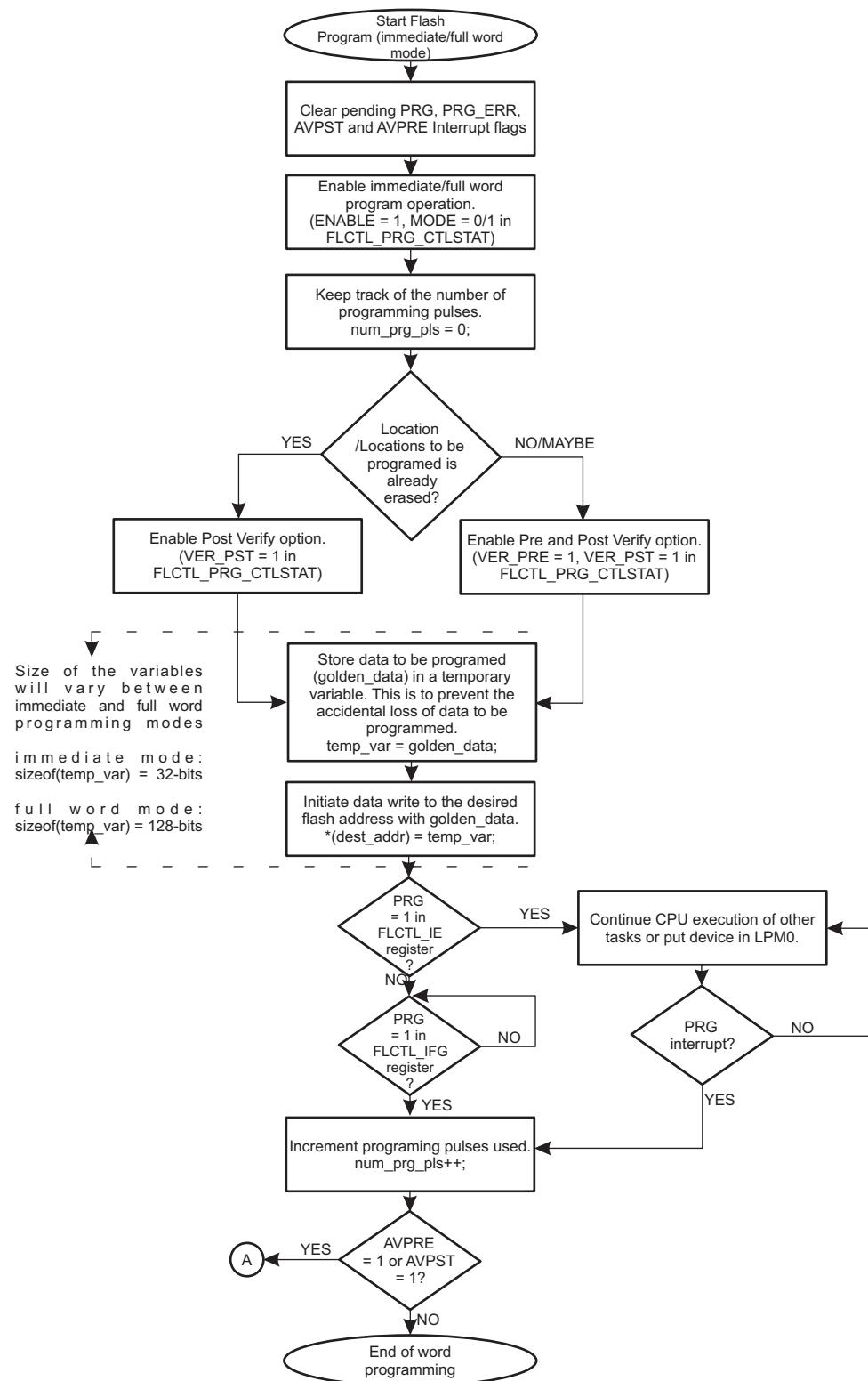
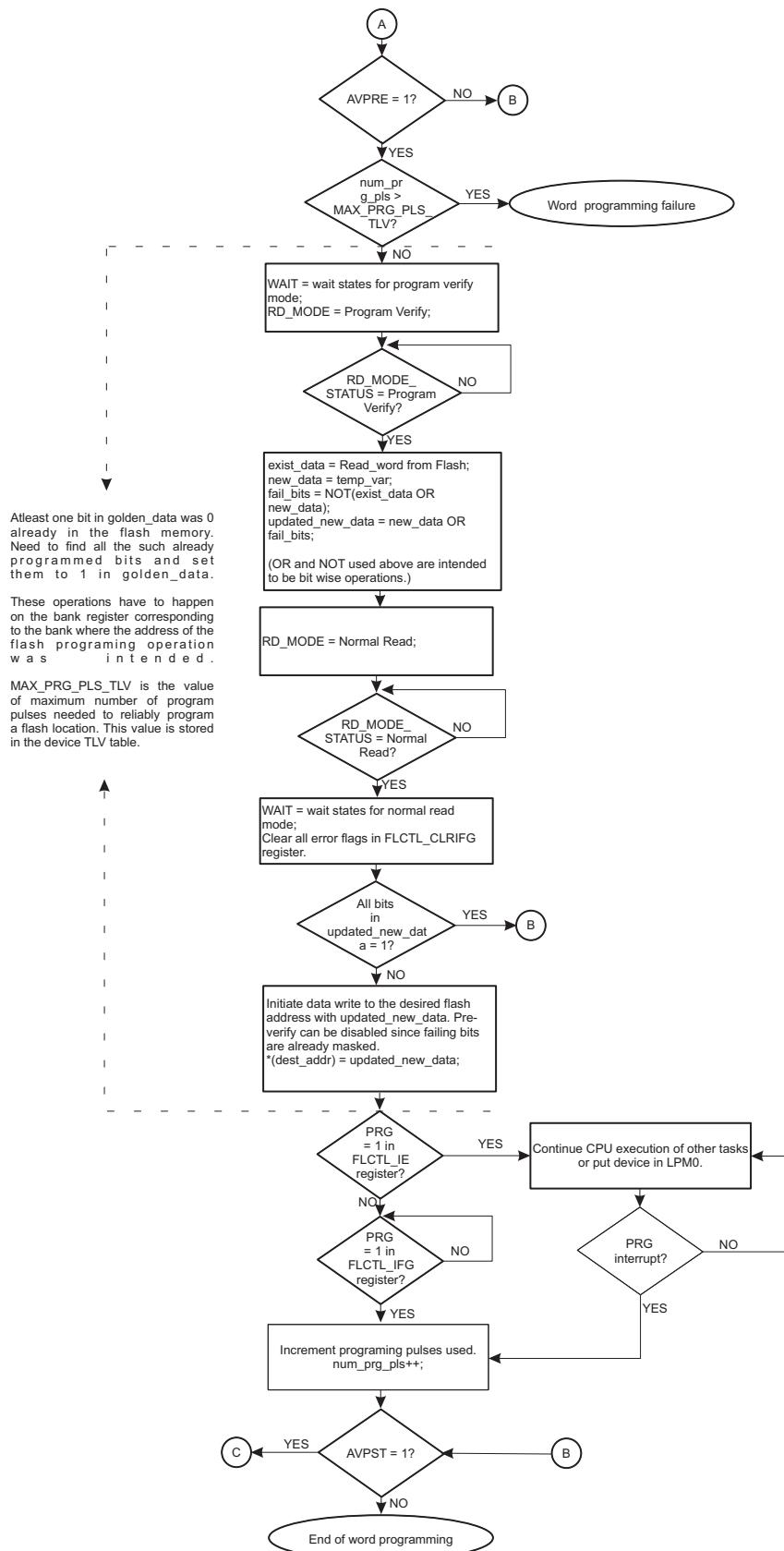
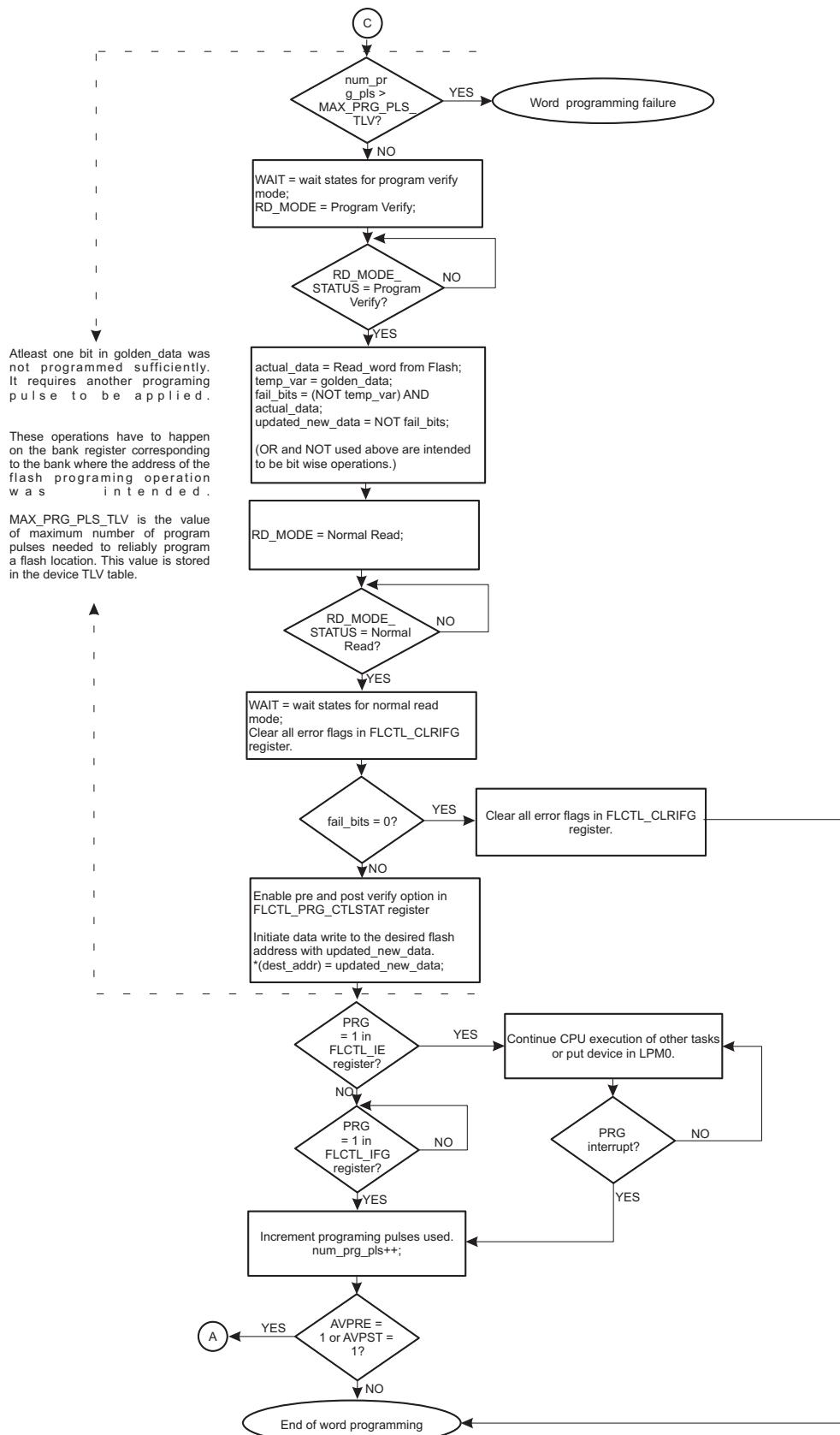


Figure 8-1. Immediate and Full Word Program Flow


**Figure 8-2. Pre-Verify Error Handling for Immediate and Full Word Program Flow**


**Figure 8-3. Post-Verify Error Handling for Immediate and Full Word Program Flow**

### 8.3.2.3 Burst Program Mode

The Burst Program feature enhances the full-word programming mode of operation by permitting multiple (up to 4) 128-bit words to be written in one single burst command. This is a valuable feature in cases where a large number of bytes (a block of data) are required to be programmed to contiguous addresses in the Flash in quick succession. Implementing a burst operation permits a lower overall write latency because the setup and hold times associated with Flash programming operation (transparently handled by the controller) are not repeated for each iteration.

To enable Burst Programming the following information needs to be made available:

- Data Input Buffer: This is a 4-deep 128-bit wide buffer that can be preloaded with data either by the application code, or through the DMA. The buffer is implemented as 16- or 32-bit registers to facilitate direct writes from the CPU and DMA. The FLCTL\_PRG\_BRST\_DATAx\_y registers are used to load the burst data.
- Start address register: The start address is configured in the FLCTL\_PRG\_BRST\_STARTADDR register (must be a 128 bit boundary)
- Burst program length: This is configured by setting the LEN bits in the FLCTL\_PRG\_BRST\_CTLSTAT to indicate the number of 128-bit words to be written in sequence (up to 4 words can be written in succession)

In this mode of operation, if all the writes are within the same sector, the setup and hold delays of the program operation are incurred only once through the burst. If the write happens to cross a sector boundary, the operation is broken into two bursts.

It is the responsibility of the application to ensure that the data input buffer registers are loaded with appropriate data to be programmed before the burst operation is started. Bits that are not to be programmed should be set to 1 to mask them from the program pulse.

Considerations for using the auto-verify feature in burst programming mode are described in [Section 8.3.2.1](#)

The Burst Program feature is configured by the ENABLE bit in the FLCTL\_PRG\_CTLSTAT register.

#### 8.3.2.3.1 Software Flow for Burst Programming Mode

[Figure 8-4](#), [Figure 8-5](#), and [Figure 8-6](#) show the software flow to be followed for reliable burst programming mode.

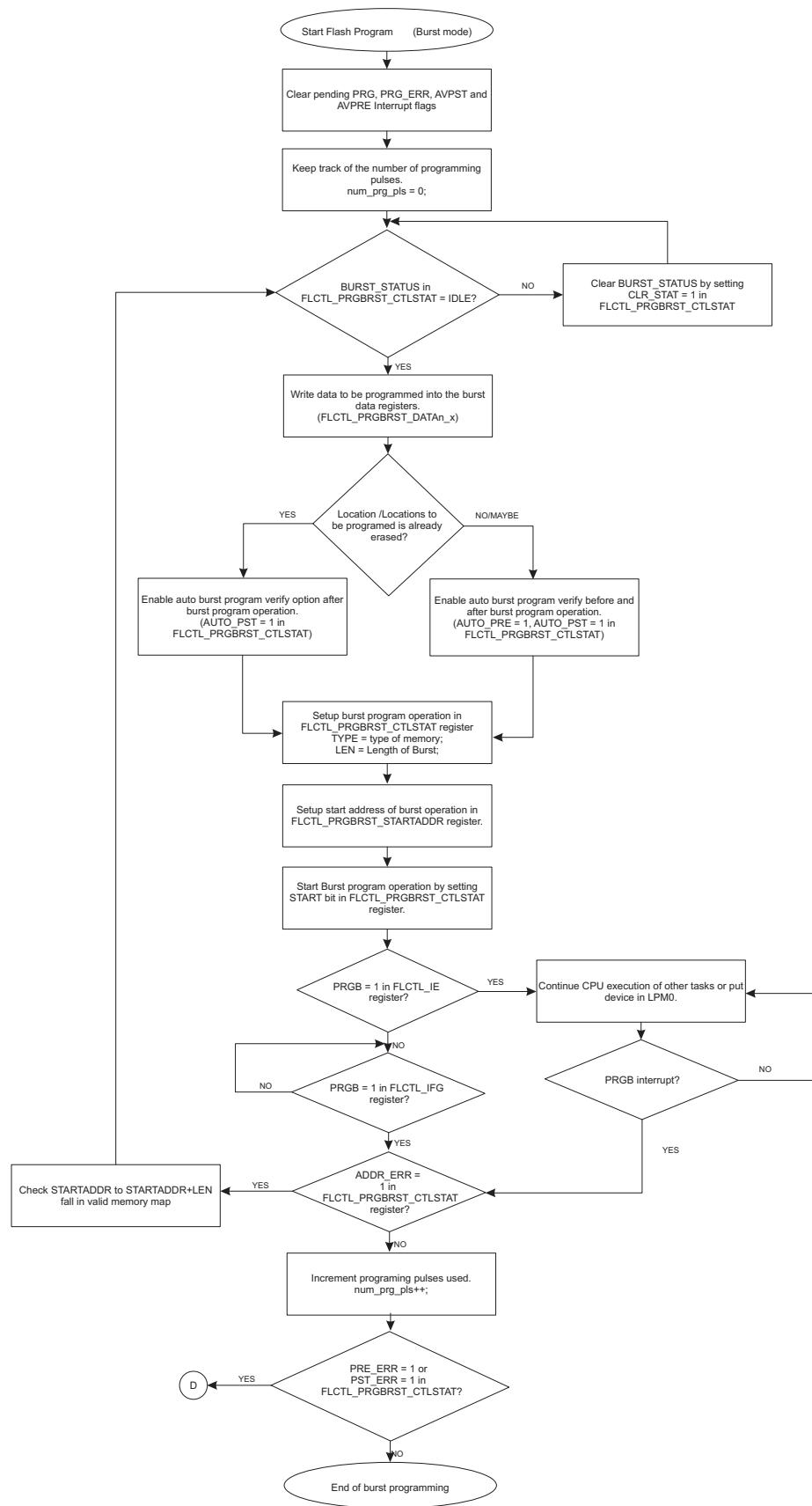
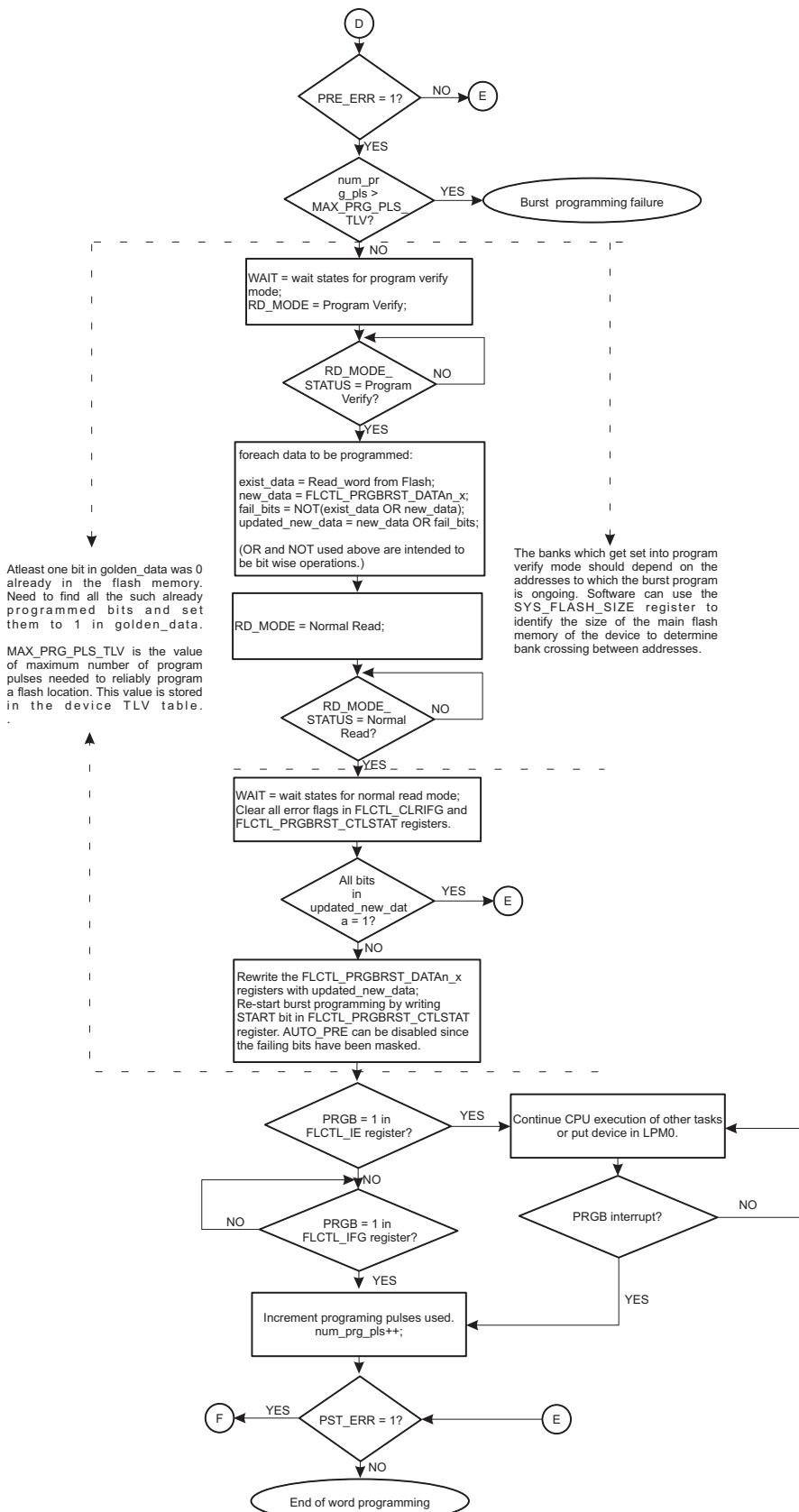
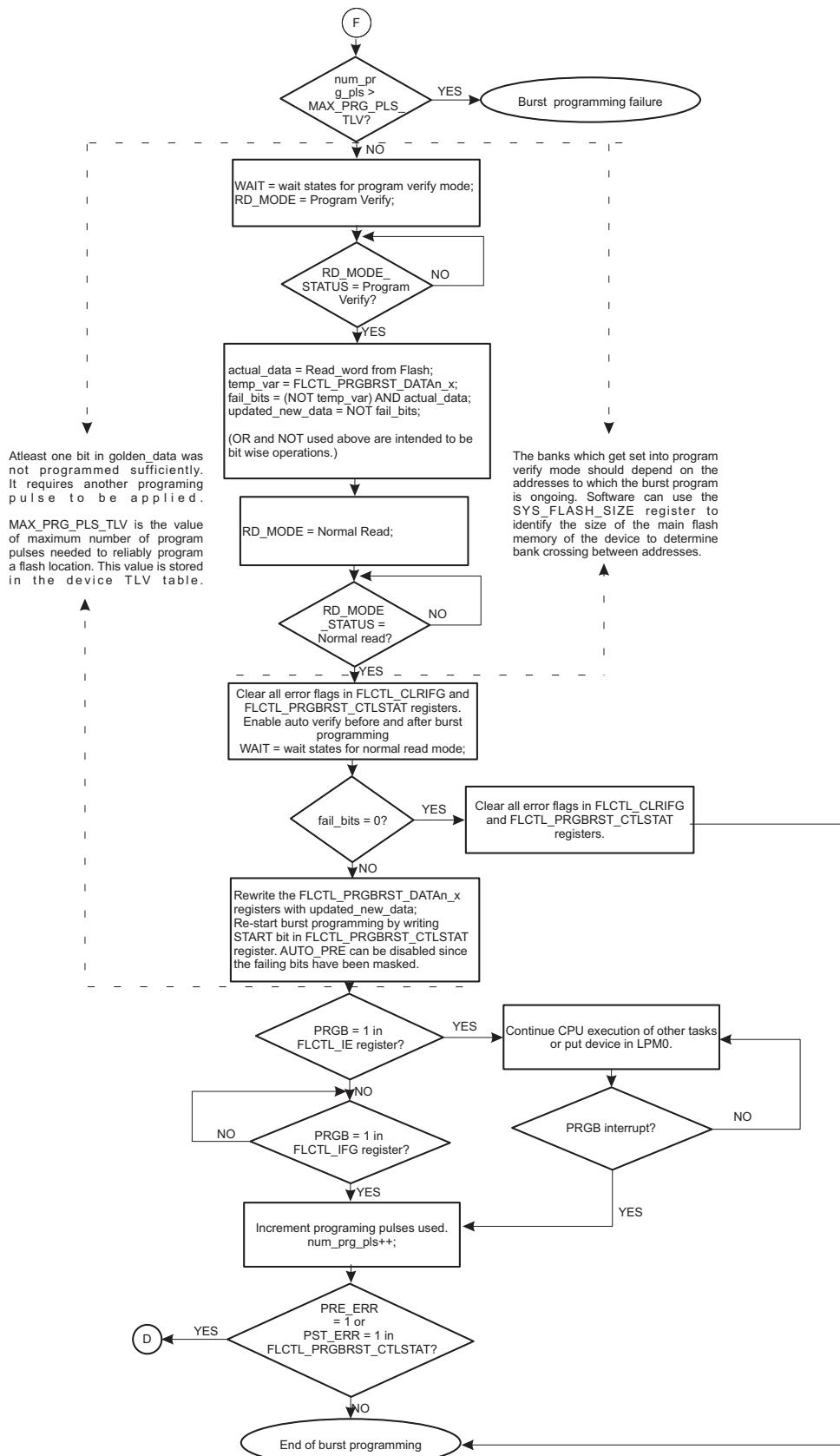


Figure 8-4. Burst Program Flow


**Figure 8-5. Handling Auto-Verify Error Before the Burst Operation**


**Figure 8-6. Handling Auto-Verify Error After the Burst Operation**

### 8.3.3 Advanced Flash Erase

Erasing MSP432 flash requires the following stages to be incorporated into user software:

1. Initiate Erase
2. Post-verify

The MSP432 Flash controller requires that user application implements the exact routines listed in [Section 8.3.3.1](#). Failure to follow the documented software flow could result in the MSP432 Flash being erased incorrectly. TI recommends that users read and understand the requirements for custom in-system programming and erasing before implementation.

An erase operation must be followed by a read of the erased locations to confirm that erase was successful. Verification of erase operation must be initiated by the user application and hardware support is not available for automatic verification. The auto-verify feature described in [Section 8.3.2.1](#) is only available when programming.

During the verification of erase operation, flash reads must be done by setting the bank in Erase-Verify-Read mode by using RD\_MODE bits in the FLCTL\_BANKx\_RDCTL register.

Additionally, because the Flash controller is optimized to provide energy and time-efficient erase operations, user application software may be required to execute the two stages of erasing multiple times, thereby subjecting the flash memory to multiple erase "pulses".

An efficient software implementation of erase verification should use the burst read and compare feature of the FLCTL. Refer to the example on erase verify in [Section 8.3.3.2](#). If burst read gives an error, then re-initiate erase until the maximum number of erase tries is met.

Each device supports a maximum number of erase pulses that is defined by the Flash Maximum Erase Pulses parameter in the device data sheet, Device Descriptors (TLV) section.

The following Flash erase modes are supported:

- Sector erase
- Mass erase

In both sector and mass erase modes, Driver Library APIs are available that implement the recommended erase flow. These APIs are also categorized into blocking and nonblocking, and the user can select whether to use the blocking version, which includes the erase verify stage, or the nonblocking version, which simply initiates the erase and relies on the user code to perform the verification. [Table 8-11](#) lists the available MSP432 driver library functions for this purpose.

**Table 8-11. MSP432 Driver Library API for Flash Erase Operations**

MSP432 Driver Library API	Function
FlashCtl_eraseSector	Erases a sector of MAIN or INFO flash memory. This is a blocking API.
FlashCtl_initiateSectorErase	Initiates a sector erase of MAIN or INFO flash memory. Note that this function simply initiates the sector erase, but does no verification which is required by the flash controller. This is a nonblocking API.
FlashCtl_performMassErase	Performs a mass erase on all unprotected flash sectors. This is a blocking API.
FlashCtl_initiateMassErase	Initiates a mass erase and returns control back to the program. This is a nonblocking API.

#### 8.3.3.1 Software Flow for Flash Erase

The below flow must be followed for any erase operation into the Flash Memory.

- An erase operation must be followed by a read of the erased locations to confirm that erase was successful.
- Verification of erase operation is to be done in software and no hardware support is available for automatic verification (unlike program operation).
- The read for verification of erase operation must be done by setting the bank in Erase Verify read mode.
- An efficient software implementation of erase verification should use the burst read and compare feature of the FLCTL. See the example on erase verify in [Section 8.3.3.2](#).

- If burst read gives an error, then re-initiate erase until the verification pass.
- On erase failure in the above attempts, software should repeat the above steps only till the maximum value of erase pulses as specified in the device TLV. The details of the TLV can be found in the device data sheet.

---

**NOTE:** **Interrupt handling during program or erase:** Application must ensure that during an active program or erase operation, interrupts are not serviced from the flash bank where program or erase is ongoing. Application can either choose to disable interrupts during the program or erase operation or use SRAM or other flash bank for any potential interrupt handling.

**Constraints for flash bank read modes before performing flash program or erase:**

Application must ensure that the flash program or erase operations are started only when the flash bank, to which the program or erase is intended, is in a normal read mode of operation.

---

### 8.3.3.2 Using Burst Read and Compare Feature for Erase Verify

This section gives an example of how the Flash Controller's Burst Read and Compare feature can be used to verify the erase operation after the erase of 1 information memory sector on the device. The sector assumed is Information memory Bank 0 Sector 0.

A pseudo code for this operation is given below:

1. Set Bank 0 in the wait-state corresponding to erase verify mode of operation.
2. Set Bank 0 in erase verify mode of operation using RD\_MODE field of the FLCTL\_BANK0\_RDCTL register.
3. Poll on the RD\_MODE\_STATUS field of the FLCTL\_BANK0\_RDCTL register to ensure that the bank 0 is in the erase verify mode.
4. Program FLCTL\_RDBRST\_STARTADDR with 0 (start address of sector 0 from the offset of the Information memory space).
5. Program FLCTL\_RDBRST\_LEN register with 4096 (one information flash sector size).
6. Program FLCTL\_RDBRST\_CTLSTAT register with the following fields:
  - (a) MEM\_TYPE: Information memory
  - (b) DATA\_CMP: 1b corresponding to a data of FFFF\_FFFF\_FFFF\_FFFF\_FFFF\_FFFF\_FFFF\_FFFF
  - (c) STOP\_FAIL: Set this bit to stop on the first compare mismatch
  - (d) START: Start the burst read operation.
7. Poll on BRST\_STAT field of the FLCTL\_RDBRST\_CTLSTAT register to ensure that the Read Burst and Compare operation is completed.
8. Check the values of the ADDR\_ERR and CMP\_ERR to ensure that the operation completed without any errors.
9. In case of errors:
  - (a) The result of the last failed compare error is reflected in the FAIL\_ADDRESS field of the FLCTL\_RDBRST\_FAILADDR register.
  - (b) The result of total number of failures encountered is reflected as a part of FLCTL\_RDBRST\_FAILCNT register.

### 8.3.4 Flash Controller Interrupts

The Flash Controller can be configured to flag (and generate interrupts on) the following exception conditions

- Completion of burst mode program operation (PRGB)
- Pre-program auto-verify error (AVPRE)
- Post-program auto-verify error (AVPST)
- Completion of immediate/full-word program operation (PRG)
- Loss of data due to erroneous writes in Full Word Program mode (explained in [Section 8.3.2.2](#))

(PRG\_ERR)

- Completion of erase operation (ERASE)
- Completion of burst read and compare operation (can also stop due to compare mismatch) (RDBRST)
- Benchmark counter match event (BMRK)

To generate an interrupt, users should enable the Flash controller interrupt at NVIC level (Refer device datasheet: **NVIC Interrupts** table) and the required interrupt enable conditions mentioned above.

---

**NOTE:** The Flash Controller generates an active interrupt for any of the enabled conditions. It is the responsibility of software to process and clear outstanding interrupt flags.

---

### 8.3.5 Application Benchmarking Features

The Flash Controller offers two counters for application benchmarking purposes. This feature is extremely useful in scenarios where monitoring the number of Flash accesses is a care-about, primarily because the Flash access power is the main contributor to the overall active power of the device. These counters are described below

- Instruction Fetch Benchmark Counter (32 bits)
  - Readable/Writable by software
  - Increments on each Instruction fetch to the Flash
- Data Fetch Benchmark Counter (32 bits)
  - Readable/Writable by software
  - Increments on each Data fetch to the Flash

In addition, the Flash Controller also implements a compare based interrupt generation capability. The interrupt generation logic can be configured to monitor either of the benchmark counters and generate an event when the counter in consideration reaches a particular value.

### 8.3.6 Support for AM\_LF\_VCOREx and LPM0\_LF\_VCOREx Power Modes

The MSP432P4xx family of devices support low-frequency active (AM\_LF\_VCOREx) and low-frequency LPM0 (LPM0\_LF\_VCOREx) modes. In these modes, the device runs in a very low-frequency low-leakage mode, with the maximum bus clock frequency restricted to 128 kHz.

When operating in this mode, the Flash Controller has the following functionality:

- Reads are carried out in normal read mode only. The Read Mode setting for both banks is automatically set to normal read when the Flash Controller detects the low-frequency active or low-frequency LPM0 modes (AM\_LF\_VCOREx or LPM0\_LF\_VCOREx modes).
- Read burst and compare operation is not permitted.
- Any form of program or erase operation is not permitted.

---

**NOTE:** It is the responsibility of the application to ensure that only read operations are carried out to the Flash when the device is in low-frequency active or low-frequency LPM0 modes. All other operations are ignored without generating an exception. If the Flash is enabled for full word write mode, and an entry to low-frequency active or low-frequency LPM0 mode is initiated, any partially composed data in the 128-bit write word is discarded without generating an exception.

---

### 8.3.7 Flash Functionality During Resets

This section concentrates on the effect of system/device level resets on the Flash Functionality.

#### 8.3.7.1 Soft Reset (Class 3)

A Soft Reset has no impact on the Flash Controller functionality.

- Any read, program, or erase operation currently being serviced continues.
- Outstanding Flash operations are processed as normal.
- New accesses or operations are processed as normal.

### 8.3.7.2 Hard Reset or POR Reset (Class 2)

A Hard Reset has the following impact on Flash Controller functionality:

- All current and outstanding read operations are terminated.
- All current and outstanding program or erase operations are terminated.
- All application settings in the Flash Control registers are reset.

## 8.4 FLCTL Registers

**Table 8-12** lists the registers of the flash controller with their address offsets. See the device-specific data sheet for the base address of the module.

**Table 8-12. FLCTL Registers**

Offset	Acronym	Register Name	Section
0x000	FLCTL_POWER_STAT	Power Status Register	<a href="#">Section 8.4.1</a>
010h	FLCTL_BANK0_RDCTL	Bank0 Read Control Register	<a href="#">Section 8.4.2</a>
014h	FLCTL_BANK1_RDCTL	Bank1 Read Control Register	<a href="#">Section 8.4.3</a>
020h	FLCTL_RDBRST_CTLSTAT	Read Burst/Compare Control and Status Register	<a href="#">Section 8.4.4</a>
024h	FLCTL_RDBRST_STARTADDR	Read Burst/Compare Start Address Register	<a href="#">Section 8.4.5</a>
028h	FLCTL_RDBRST_LEN	Read Burst/Compare Length Register	<a href="#">Section 8.4.6</a>
03Ch	FLCTL_RDBRST_FAILADDR	Read Burst/Compare Fail Address Register	<a href="#">Section 8.4.7</a>
040h	FLCTL_RDBRST_FAILCNT	Read Burst/Compare Fail Count Register	<a href="#">Section 8.4.8</a>
050h	FLCTL_PRG_CTLSTAT	Program Control and Status Register	<a href="#">Section 8.4.9</a>
054h	FLCTL_PRGBRST_CTLSTAT	Program Burst Control and Status Register	<a href="#">Section 8.4.10</a>
058h	FLCTL_PRGBRST_STARTADDR	Program Burst Start Address Register	<a href="#">Section 8.4.11</a>
060h	FLCTL_PRGBRST_DATA0_0	Program Burst Data0 Register0	<a href="#">Section 8.4.12</a>
064h	FLCTL_PRGBRST_DATA0_1	Program Burst Data0 Register1	<a href="#">Section 8.4.13</a>
068h	FLCTL_PRGBRST_DATA0_2	Program Burst Data0 Register2	<a href="#">Section 8.4.14</a>
06Ch	FLCTL_PRGBRST_DATA0_3	Program Burst Data0 Register3	<a href="#">Section 8.4.15</a>
070h	FLCTL_PRGBRST_DATA1_0	Program Burst Data1 Register0	<a href="#">Section 8.4.16</a>
074h	FLCTL_PRGBRST_DATA1_1	Program Burst Data1 Register1	<a href="#">Section 8.4.17</a>
078h	FLCTL_PRGBRST_DATA1_2	Program Burst Data1 Register2	<a href="#">Section 8.4.18</a>
07Ch	FLCTL_PRGBRST_DATA1_3	Program Burst Data1 Register3	<a href="#">Section 8.4.19</a>
080h	FLCTL_PRGBRST_DATA2_0	Program Burst Data2 Register0	<a href="#">Section 8.4.20</a>
084h	FLCTL_PRGBRST_DATA2_1	Program Burst Data2 Register1	<a href="#">Section 8.4.21</a>
088h	FLCTL_PRGBRST_DATA2_2	Program Burst Data2 Register2	<a href="#">Section 8.4.22</a>
08Ch	FLCTL_PRGBRST_DATA2_3	Program Burst Data2 Register3	<a href="#">Section 8.4.23</a>
090h	FLCTL_PRGBRST_DATA3_0	Program Burst Data3 Register0	<a href="#">Section 8.4.24</a>
094h	FLCTL_PRGBRST_DATA3_1	Program Burst Data3 Register1	<a href="#">Section 8.4.25</a>
098h	FLCTL_PRGBRST_DATA3_2	Program Burst Data3 Register2	<a href="#">Section 8.4.26</a>
09Ch	FLCTL_PRGBRST_DATA3_3	Program Burst Data3 Register3	<a href="#">Section 8.4.27</a>
0A0h	FLCTL_ERASE_CTLSTAT	Erase Control and Status Register	<a href="#">Section 8.4.28</a>
0A4h	FLCTL_ERASE_SECTADDR	Erase Sector Address Register	<a href="#">Section 8.4.29</a>
0B0h	FLCTL_BANK0_INFO_WEPROT	Information Memory Bank0 Write/Erase Protection Register	<a href="#">Section 8.4.30</a>
0B4h	FLCTL_BANK0_MAIN_WEPROT	Main Memory Bank0 Write/Erase Protection Register	<a href="#">Section 8.4.31</a>
0C0h	FLCTL_BANK1_INFO_WEPROT	Information Memory Bank1 Write/Erase Protection Register	<a href="#">Section 8.4.32</a>
0C4h	FLCTL_BANK1_MAIN_WEPROT	Main Memory Bank1 Write/Erase Protection Register	<a href="#">Section 8.4.33</a>
0D0h	FLCTL_BMRK_CTLSTAT	Benchmark Control and Status Register	<a href="#">Section 8.4.34</a>
0D4h	FLCTL_BMRK_IFETCH	Benchmark Instruction Fetch Count Register	<a href="#">Section 8.4.35</a>
0D8h	FLCTL_BMRK_DREAD	Benchmark Data Read Count Register	<a href="#">Section 8.4.36</a>
0DCh	FLCTL_BMRK_CMP	Benchmark Count Compare Register	<a href="#">Section 8.4.37</a>
0F0h	FLCTL_IFG	Interrupt Flag Register	<a href="#">Section 8.4.38</a>
0F4h	FLCTL_IE	Interrupt Enable Register	<a href="#">Section 8.4.39</a>
0F8h	FLCTL_CLRIFG	Clear Interrupt Flag Register	<a href="#">Section 8.4.40</a>
0FCh	FLCTL_SETIFG	Set Interrupt Flag Register	<a href="#">Section 8.4.41</a>
100h	FLCTL_READ_TIMCTL	Read Timing Control Register	<a href="#">Section 8.4.42</a>

**Table 8-12. FLCTL Registers (continued)**

Offset	Acronym	Register Name	Section
104h	FLCTL_READMARGIN_TIMCTL	Read Margin Timing Control Register	<a href="#">Section 8.4.43</a>
108h	FLCTL_PRGVER_TIMCTL	Program Verify Timing Control Register	<a href="#">Section 8.4.44</a>
10Ch	FLCTL_ERSVER_TIMCTL	Erase Verify Timing Control Register	<a href="#">Section 8.4.45</a>
114h	FLCTL_PROGRAM_TIMCTL	Program Timing Control Register	<a href="#">Section 8.4.46</a>
118h	FLCTL_ERASE_TIMCTL	Erase Timing Control Register	<a href="#">Section 8.4.47</a>
11Ch	FLCTL_MASSERASE_TIMCTL	Mass Erase Timing Control Register	<a href="#">Section 8.4.48</a>
120h	FLCTL_BURSTPRG_TIMCTL	Burst Program Timing Control Register	<a href="#">Section 8.4.49</a>

---

**NOTE:** This is a 32-bit module and can be accessed through word (32-bit) or half-word (16-bit) or byte (8-bit) accesses.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### **8.4.1 FLCTL\_POWER\_STAT Register (offset = 0000h)**

Flash Power Status Register

**Figure 8-7. FLCTL\_POWER\_STAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
RD_2T	TRIMSTAT	IREFSTAT	VREFSTAT	LDOSTAT	PSTAT										
r	r	r	r	r	r	r	r	r	r-<1>	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)	r-(0)

**Table 8-13. FLCTL\_POWER\_STAT Register Description**

Bit	Field	Type	Reset	Description
31-8	Reserved	R	NA	Reserved. Reads return 0h
7	RD_2T	R	1h	Indicates if Flash is being accessed in 2T mode 0b = Flash reads are in 1T mode 1b = Flash reads are in 2T mode
6	TRIMSTAT	R	0h	PSS trim done status 0b = PSS trim not complete 1b = PSS trim complete
5	IREFSTAT	R	0h	PSS IREF stable status 0b = IREF not stable 1b = IREF stable
4	VREFSTAT	R	0h	PSS VREF stable status 0b = Flash LDO not stable 1b = Flash LDO stable
3	LDOSTAT	R	0h	PSS FLDO GOOD status 0b = FLDO not GOOD 1b = FLDO GOOD
2-0	PSTAT	R	0h	Flash power status 000b = Flash IP in power-down mode 001b = Flash IP Vdd domain power-up in progress 010b = PSS LDO_GOOD, IREF_OK and VREF_OK check in progress 011b = Flash IP SAFE_LV check in progress 100b = Flash IP Active 101b = Flash IP Active in Low-Frequency Active and Low-Frequency LPM0 modes. 110b = Flash IP in Standby mode 111b = Flash IP in Current mirror boost state

### 8.4.2 FLCTL\_BANK0\_RDCTL Register (offset = 0010h)

Flash Bank0 Read Control Register

**Figure 8-8. FLCTL\_BANK0\_RDCTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												RD_MODE_STATUS			
r	r	r	r	r	r	r	r	r	r	r	r	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAIT				Reserved				Reserved		BUFD	BUFI	RD_MODE			
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r	r	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 8-14. FLCTL\_BANK0\_RDCTL Register Description**

Bit	Field	Type	Reset	Description
31-20	Reserved	R	NA	Reserved. Reads return 0h
19-16	RD_MODE_STATUS <sup>(1)</sup>	R	0h	Reflects the current Read Mode of the Bank 0000b = Normal read mode 0001b = Read Margin 0 0010b = Read Margin 1 0011b = Program Verify 0100b = Erase Verify All others = Reserved
15-12	WAIT <sup>(2)(3)(4)</sup>	RW	0h	Defines the number of wait states required for a read operation to the bank 0000b = 0 wait states 0001b = 1 wait states 0010b = 2 wait states 0011b = 3 wait states 0100b = 4 wait states 0101b = 5 wait states 0110b = 6 wait states 0111b = 7 wait states 1000b = 8 wait states 1001b = 9 wait states 1010b = 10 wait states 1011b = 11 wait states 1100b = 12 wait states 1101b = 13 wait states 1110b = 14 wait states 1111b = 15 wait states
11-8	Reserved	RW	0h	Reserved
7-6	Reserved	R	NA	Reserved. Reads return 0h
5	BUFD	RW	0h	Enables read buffering feature for data reads to this bank
4	BUFI	RW	0h	Enables read buffering feature for instruction fetches to this bank

(1) These bits are forced to 0h when the device is in Low-Frequency Active and Low-Frequency LPM0 modes of operation.

(2) The number of wait states required for read depends on *both* the bus clock frequency and the mode of read. Refer to the device electrical characteristics for details on the wait state parameters

(3) If the bus clock frequency is being changed to a higher value, it is the application's responsibility to ensure that the wait state value is changed before the frequency change is effected. If this is not followed, the device behavior is not deterministic.

(4) This bit field is writable **only** when burst status (17:16) of the FLCTL\_RDBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

**Table 8-14. FLCTL\_BANK0\_RDCTL Register Description (continued)**

Bit	Field	Type	Reset	Description
3-0	RD_MODE <sup>(1)(5)(4)</sup>	RW	0h	Flash read mode control setting for Bank 0000b = Normal read mode 0001b = Read Margin 0 0010b = Read Margin 1 0011b = Program Verify 0100b = Erase Verify All others = Reserved

<sup>(5)</sup> These bits are forced to 0h when the device is in 2T mode of operation.

### 8.4.3 FLCTL\_BANK1\_RDCTL Register (offset = 0014h)

Flash Bank1 Read Control Register

**Figure 8-9. FLCTL\_BANK1\_RDCTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												RD_MODE_STATUS			
r	r	r	r	r	r	r	r	r	r	r	r	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAIT				Reserved				Reserved		BUFD	BUFI	RD_MODE			
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r	r	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 8-15. FLCTL\_BANK1\_RDCTL Register Description**

Bit	Field	Type	Reset	Description
31-20	Reserved	R	NA	Reserved. Reads return 0h
19-16	RD_MODE_STATUS <sup>(1)</sup>	R	0h	Reflects the current Read Mode of the bank 0000b = Normal read mode 0001b = Read Margin 0 0010b = Read Margin 1 0011b = Program Verify 0100b = Erase Verify All others = Reserved
15-12	WAIT <sup>(2)(3)(4)</sup>	RW	0h	Defines the number of wait states required for a read operation to the bank 0000b = 0 wait states 0001b = 1 wait states 0010b = 2 wait states 0011b = 3 wait states 0100b = 4 wait states 0101b = 5 wait states 0110b = 6 wait states 0111b = 7 wait states 1000b = 8 wait states 1001b = 9 wait states 1010b = 10 wait states 1011b = 11 wait states 1100b = 12 wait states 1101b = 13 wait states 1110b = 14 wait states 1111b = 15 wait states
11-8	Reserved	RW	0h	Reserved
7-6	Reserved	R	NA	Reserved. Reads return 0h
5	BUFD	RW	0h	Enables read buffering feature for data reads to this bank
4	BUFI	RW	0h	Enables read buffering feature for instruction fetches to this bank

(1) These bits are set to 0h when the device is in Low-Frequency Active and Low-Frequency LPM0 modes of operation

(2) Number of wait states required for read are dependent on *both* the bus clock frequency and the mode of read. Refer to the device electrical characteristics for details on the wait state parameters

(3) If the bus clock frequency is being changed to a higher value, it is the application's responsibility to ensure that the wait state value is changed before the frequency change is effected. If this is not followed, the device behavior is not deterministic.

(4) This bit field is writable **only** when burst status (17:16) of the FLCTL\_RDBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

**Table 8-15. FLCTL\_BANK1\_RDCTL Register Description (continued)**

Bit	Field	Type	Reset	Description
3-0	RD_MODE <sup>(1)(5)(4)</sup>	RW	0h	Flash read mode control setting for the bank 0000b = Normal read mode 0001b = Read Margin 0 0010b = Read Margin 1 0011b = Program Verify 0100b = Erase Verify All others = Reserved

<sup>(5)</sup> These bits are forced to 0h when the device is in 2T mode of operation.

#### 8.4.4 FLCTL\_RDBRST\_CTLSTAT Register (offset = 0020h)

Flash Read Burst/Compare Control and Status Register

**Figure 8-10. FLCTL\_RDBRST\_CTLSTAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							CLR_STAT	Reserved			ADDR_ERR	CMP_ERR	BRST_STAT		
r	r	r	r	r	r	r	w	r	r	r	r	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DATA_CMP	STOP_FAIL	MEM_TYPE	STAR_T					
r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0	w-0

**Table 8-16. FLCTL\_RDBRST\_CTLSTAT Register Description**

Bit	Field	Type	Reset	Description
31-24	Reserved	R	NA	Reserved. Reads return 0h
23	CLR_STAT <sup>(1)</sup>	W	NA	Write 1 to clear status bits 19-16 of this register Write '0' has no effect
22-20	Reserved	R	NA	Reserved. Reads return 0h
19	ADDR_ERR	R	0h	If 1, indicates that Burst/Compare Operation was terminated due to access to reserved memory
18	CMP_ERR	R	0h	if 1, indicates that the Burst/Compare Operation encountered at least one data comparison error
17-16	BRST_STAT	R	0h	Status of Burst/Compare operation 00b = Idle 01b = Burst/Compare START bit written, but operation pending 10b = Burst/Compare in progress 11b = Burst complete (status of completed burst remains in this state unless explicitly cleared by software)
15-6	Reserved	R	NA	Reserved. Reads return 0h
5	Reserved	RW	0h	Reserved
4	DATA_CMP <sup>(2)</sup>	RW	0h	Data pattern used for comparison against memory read data 0b = 0000_0000_0000_0000_0000_0000_0000 1b = FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF
3	STOP_FAIL <sup>(2)</sup>	RW	0h	If set to 1, causes burst/compare operation to terminate on first compare mismatch
2-1	MEM_TYPE <sup>(2)</sup>	RW	0h	Type of memory that burst is carried out on 00b = Main Memory 01b = Information memory 10b = Reserved 11b = Reserved
0	START <sup>(3)(2)</sup>	W	0h	Write 1 triggers start of burst/compare operation

<sup>(1)</sup> Write 1 to CLRSTAT clears the status bits 19:16 **only** when burst status (17:16) shows completion state. In all other cases, write-1 has no effect. This is to allow deterministic behavior.

<sup>(2)</sup> This bit field is writable **only** when burst status (17:16) of the FLCTL\_RDBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

<sup>(3)</sup> Writes to the START bit are ignored if the device is in Low-Frequency Active and Low-Frequency LPM0 modes of operation

#### **8.4.5 FLCTL\_RDBRST\_STARTADDR Register (offset = 0024h)**

Flash Read Burst/Compare Start Address Register

**Figure 8-11. FLCTL\_RDBRST\_STARTADDR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										START_ADDRESS					
r	r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
START_ADDRESS															
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r	r	r	r

**Table 8-17. FLCTL\_RDBRST\_STARTADDR Register Description**

Bit	Field	Type	Reset	Description
31-21	Reserved	R	0h	Reserved. Reads return 0h
20-0	START_ADDRESS <sup>(1)(2)</sup>	RW (with exception s)	0h	Start Address of Burst Operation. Offset from 0h, with 0h as start address of the type of memory region selected Bits 3-0 are always 0 (forced 128bit boundary)

<sup>(1)</sup> If the amount of memory available is less than 2MB, the upper bits of the START\_ADDRESS behave as reserved. To know the amount of Flash memory available, refer to the device data sheet

<sup>(2)</sup> This bit field is writable **only** when burst status (17:16) of the FLCTL\_RDBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.6 FLCTL\_RDBRST\_LEN Register (offset = 0028h)

Flash Read Burst/Compare Length Register

**Figure 8-12. FLCTL\_RDBRST\_LEN Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										BURST_LENGTH					
r	r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BURST_LENGTH															
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r	r	r	r

**Table 8-18. FLCTL\_RDBRST\_LEN Register Description**

Bit	Field	Type	Reset	Description
31-21	Reserved	R	NA	Reserved. Reads return 0h
20-0	BURST_LENGTH <sup>(1)(2)</sup>	RW (with exception s)	0h	Length of Burst Operation (in bytes). Bits 3-0 are always 0 (forced minimum resolution of 128bits)

<sup>(1)</sup> If amount of memory available is less than 2MB, the upper bits of the BURST\_LENGTH behave as reserved. To know actual amount of Flash memory available, refer to the device data sheet

<sup>(2)</sup> This bit field is writable **only** when burst status (17:16) of the FLCTL\_RDBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

### **8.4.7 FLCTL\_RDBRST\_FAILADDR Register (offset = 003Ch)**

Flash Read Burst/Compare Fail Address Register

**Figure 8-13. FLCTL\_RDBRST\_FAILADDR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										FAIL_ADDR					
r	r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_ADDRESS															
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r	r	r	r

**Table 8-19. FLCTL\_RDBRST\_FAILADDR Register Description**

Bit	Field	Type	Reset	Description
31-21	Reserved	R	NA	Reserved. Reads return 0h
20-0	FAIL_ADDRESS <sup>(1)(2)(3)</sup>	RW (with exception s)	0h	Reflects address of last failed compare. Offset from 0h, with 0h as start address of the type of memory region selected Bits 3-0 are always 0 (forced 128bit boundary)

- <sup>(1)</sup> Application may choose to clear this register to 0h before starting a new burst compare operation
- <sup>(2)</sup> If amount of memory available is less than 2MB, the upper bits of the FAIL\_ADDR behave as reserved. To know actual amount of Flash memory available, refer to the device data sheet
- <sup>(3)</sup> This bit field is writable **only** when burst status (17:16) of the FLCTL\_RDBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

### 8.4.8 FLCTL\_RDBRST\_FAILCNT Register (offset = 0040h)

Flash Read Burst/Compare Fail Count Register

**Figure 8-14. FLCTL\_RDBRST\_FAILCNT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															FAIL_COUN T
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_COUNT															
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 8-20. FLCTL\_RDBRST\_FAILCNT Register Description**

Bit	Field	Type	Reset	Description
31-17	Reserved	R	NA	Reserved. Reads return 0h
16-0	FAIL_COUNT <sup>(1)(2)(3)</sup>	RW	0h	Reflects number of failures encountered in burst operation.

<sup>(1)</sup> Application may choose to clear this register to 0h before starting a new burst compare operation. If the register is not cleared, it increments from the current value each time a new burst is started.

<sup>(2)</sup> FAIL\_COUNT may be as high as 128K for a 2MB Flash memory size. In case size of memory on device is less than 2MB, upper bits behave as reserved

<sup>(3)</sup> This bit field is writable **only** when burst status (17:16) of the FLCTL\_RDBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

### 8.4.9 FLCTL\_PRG\_CTLSTAT Register (offset = 0050h)

Flash Program Control and Status Register

**Figure 8-15. FLCTL\_PRG\_CTLSTAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												BNK_ACT	STATUS		
r	r	r	r	r	r	r	r	r	r	r	r	r	r-0	r-0	r-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												VER_PST	VER_PRE	MODE	ENABLE
r	r	r	r	r	r	r	r	r	r	r	r	rw-1	rw-1	rw-0	rw-0

**Table 8-21. FLCTL\_PRG\_CTLSTAT Register Description**

Bit	Field	Type	Reset	Description
31-19	Reserved	R	NA	Reserved. Reads return 0h
18	BNK_ACT	R	0h	Reflects which bank is currently undergoing a program operation (valid only if bits 17-16 don't show idle) 0b = Word in Bank0 being programmed 1b = Word in Bank1 being programmed
17-16	STATUS	R	0h	Reflects the status of program operations in the Flash memory 00b = Idle (no program operation currently active) 01b = Single word program operation triggered, but pending 10b = Single word program in progress 11b = Reserved (Idle)
15-4	Reserved	R	NA	Reserved. Reads return 0h
3	VER_PST <sup>(1)</sup>	RW	1h	Controls automatic post program verify operations 0b = No post program verification 1b = Post verify feature automatically invoked for each write operation (irrespective of the mode)
2	VER_PRE <sup>(1)</sup>	RW	1h	Controls automatic pre program verify operations 0b = No pre program verification 1b = Pre verify feature automatically invoked for each write operation (irrespective of the mode)
1	MODE <sup>(1)</sup>	RW	0h	Controls write mode selected by application 0b = Write immediate mode. Starts program operation immediately on each write to the Flash 1b = Full word write mode. Flash controller collates data over multiple writes to compose the full 128bit word before initiating the program operation <sup>(2)</sup>
0	ENABLE <sup>(3)(1)</sup>	RW	0h	Master control for all word program operations 0b = Word program operation disabled 1b = Word program operation enabled

<sup>(1)</sup> This bit field is writable **only** when burst status (17:16) of the FLCTL\_PRG\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

<sup>(2)</sup> The application must ensure that the writes in this mode follow the LSB and MSB loading order within the 128-bit address boundary. See [Section 8.3.2.2](#) for more details.

<sup>(3)</sup> This bit is forced to 0h when the device is in low-frequency active and low-frequency LPM0 modes of operation

### 8.4.10 FLCTL\_PRGBRST\_CTLSTAT Register (offset = 0054h)

Flash Program Burst Control and Status Register

**Figure 8-16. FLCTL\_PRGBRST\_CTLSTAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CLR_STAT	Reserv ed	ADDR _ERR	PST_E RR	PRE_ ERR	BURST_STATUS		
r	r	r	r	r	r	r	r	w	r	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								AUTO_PST	AUTO_PRE	LEN			TYPE	STAR T	
r	r	r	r	r	r	r	r	rw-1	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	w

**Table 8-22. FLCTL\_PRGBRST\_CTLSTAT Register Description**

Bit	Field	Type	Reset	Description
31-24	Reserved	R	NA	Reserved. Reads return 0h
23	CLR_STAT <sup>(1)</sup>	W	NA	Write 1 to clear status bits 21:16 of this register Write '0' has no effect
22	Reserved	R	NA	Reserved. Reads return 0h
21	ADDR_ERR	R	0h	If 1, indicates that Burst Operation was terminated due to attempted program of reserved memory
20	PST_ERR	R	0h	if 1, indicates that the Burst Operation encountered post-program auto-verify errors
19	PRE_ERR	R	0h	If 1, indicates that Burst Operation encountered pre-program auto-verify errors
18-16	BURST_STATUS	R	0h	At any point in time, it reflects the status of a Burst Operation 000b = Idle (Burst not active) 001b = Burst program started but pending 010b = Burst active, with 1st 128 bit word being written into Flash 011b = Burst active, with 2nd 128 bit word being written into Flash 100b = Burst active, with 3rd 128 bit word being written into Flash 101b = Burst active, with 4th 128 bit word being written into Flash 110b = Reserved (Idle) 111b = Burst Complete (status of completed burst remains in this state unless explicitly cleared by software)
15-8	Reserved	R	NA	Reserved. Reads return 0h
7	AUTO_PST <sup>(2)</sup>	RW	1h	Controls the Auto-Verify operation after the Burst Program 0b = No program verify operations carried out 1b = Causes an automatic Burst Program Verify after the Burst Program Operation
6	AUTO_PRE <sup>(2)</sup>	RW	1h	Controls the Auto-Verify operation before the Burst Program 0b = No program verify operations carried out 1b = Causes an automatic Burst Program Verify before the Burst Program Operation

(1) Write 1 to CLR\_STAT clears the status bits 21:16 **only** when burst status (18:16) shows completion state. In all other cases, write-1 has no effect. This is to allow deterministic behavior.

(2) This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

**Table 8-22. FLCTL\_PRGBRST\_CTLSTAT Register Description (continued)**

Bit	Field	Type	Reset	Description
5-3	LEN <sup>(2)</sup>	RW	0h	Length of burst (in 128 bit granularity) 000b = No burst operation 001b = 1 word burst of 128 bits, starting with address in the FLCTL_PRGBRST_STARTADDR Register 010b = 2*128 bits burst write, starting with address in the FLCTL_PRGBRST_STARTADDR Register 011b = 3*128 bits burst write, starting with address in the FLCTL_PRGBRST_STARTADDR Register 100b = 4*128 bits burst write, starting with address in the FLCTL_PRGBRST_STARTADDR Register 101b = Reserved. No burst operation. 110b = Reserved. No burst operation. 111b = Reserved. No burst operation.
2-1	TYPE <sup>(2)</sup>	RW	0h	Type of memory that burst program is carried out on 00b = Main Memory 01b = Information memory 10b = Reserved 11b = Reserved
0	START <sup>(3)(2)</sup>	W	NA	Write 1 triggers start of burst program operation

<sup>(3)</sup> Writes to the START bit are ignored if the device is in Low-Frequency Active and Low-Frequency LPM0 modes of operation

#### 8.4.11 FLCTL\_PRGBRST\_STARTADDR Register (offset = 0058h)

Flash Program Burst Start Address Register

**Figure 8-17. FLCTL\_PRGBRST\_STARTADDR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										START_ADDRESS					
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
START_ADDRESS															
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0	r-0	r-0

**Table 8-23. FLCTL\_PRGBRST\_STARTADDR Register Description**

Bit	Field	Type	Reset	Description
31-22	Reserved	R	NA	Reserved. Reads return 0h
21-0	START_ADDRESS <sup>(1)(2)</sup>	RW	0h	Start Address of Program Burst Operation. Offset from 0h, with 0h as start address of the type of memory region selected Bits 3-0 are always 0 (forced to 128-bit boundary)

<sup>(1)</sup> Start Address is set as max of 4MB for future enhancement purposes. To know actual amount of Flash memory available, refer to the device data sheet

<sup>(2)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.12 FLCTL\_PRGBRST\_DATA0\_0 Register (offset = 060h)

Flash Program Burst Data0 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-18. FLCTL\_PRGBRST\_DATA0\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 8-24. FLCTL\_PRGBRST\_DATA0\_0 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 0 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.13 FLCTL\_PRGBRST\_DATA0\_1 Register (offset = 064h)

Flash Program Burst Data0 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-19. FLCTL\_PRGBRST\_DATA0\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 8-25. FLCTL\_PRGBRST\_DATA0\_1 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 0 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.14 FLCTL\_PRGBRST\_DATA0\_2 Register (offset = 068h)

Flash Program Burst Data0 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-20. FLCTL\_PRGBRST\_DATA0\_2 Register**

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

**Table 8-26. FLCTL\_PRGBRST\_DATA0\_2 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 0 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.15 FLCTL\_PRGBRST\_DATA0\_3 Register (offset = 06Ch)

Flash Program Burst Data0 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-21. FLCTL\_PRGBRST\_DATA0\_3 Register**

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

**Table 8-27. FLCTL\_PRGBRST\_DATA0\_3 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 0 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.16 FLCTL\_PRGBRST\_DATA1\_0 Register (offset = 070h)

Flash Program Burst Data1 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-22. FLCTL\_PRGBRST\_DATA1\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 8-28. FLCTL\_PRGBRST\_DATA1\_0 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 1 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.17 FLCTL\_PRGBRST\_DATA1\_1 Register (offset = 074h)

Flash Program Burst Data1 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-23. FLCTL\_PRGBRST\_DATA1\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 8-29. FLCTL\_PRGBRST\_DATA1\_1 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 1 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.18 FLCTL\_PRGBRST\_DATA1\_2 Register (offset = 078h)

Flash Program Burst Data1 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-24. FLCTL\_PRGBRST\_DATA1\_2 Register**

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

**Table 8-30. FLCTL\_PRGBRST\_DATA1\_2 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 1 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.19 FLCTL\_PRGBRST\_DATA1\_3 Register (offset = 07Ch)

Flash Program Burst Data1 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-25. FLCTL\_PRGBRST\_DATA1\_3 Register**

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

**Table 8-31. FLCTL\_PRGBRST\_DATA1\_3 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 1 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.20 FLCTL\_PRGBRST\_DATA2\_0 Register (offset = 080h)

Flash Program Burst Data2 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-26. FLCTL\_PRGBRST\_DATA2\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 8-32. FLCTL\_PRGBRST\_DATA2\_0 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 2 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.21 FLCTL\_PRGBRST\_DATA2\_1 Register (offset = 084h)

Flash Program Burst Data2 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-27. FLCTL\_PRGBRST\_DATA2\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 8-33. FLCTL\_PRGBRST\_DATA2\_1 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 2 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.22 FLCTL\_PRGBRST\_DATA2\_2 Register (offset = 088h)

Flash Program Burst Data2 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-28. FLCTL\_PRGBRST\_DATA2\_2 Register**

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

**Table 8-34. FLCTL\_PRGBRST\_DATA2\_2 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 2 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.23 FLCTL\_PRGBRST\_DATA2\_3 Register (offset = 08Ch)

Flash Program Burst Data2 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-29. FLCTL\_PRGBRST\_DATA2\_3 Register**

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

**Table 8-35. FLCTL\_PRGBRST\_DATA2\_3 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 2 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.24 FLCTL\_PRGBRST\_DATA3\_0 Register (offset = 090h)

Flash Program Burst Data3 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-30. FLCTL\_PRGBRST\_DATA3\_0 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 8-36. FLCTL\_PRGBRST\_DATA3\_0 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 3 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.25 FLCTL\_PRGBRST\_DATA3\_1 Register (offset = 094h)

Flash Program Burst Data3 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-31. FLCTL\_PRGBRST\_DATA3\_1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DATAIN</b>															
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 8-37. FLCTL\_PRGBRST\_DATA3\_1 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 3 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.26 FLCTL\_PRGBRST\_DATA3\_2 Register (offset = 098h)

Flash Program Burst Data3 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-32. FLCTL\_PRGBRST\_DATA3\_2 Register**

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

**Table 8-38. FLCTL\_PRGBRST\_DATA3\_2 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 3 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

#### 8.4.27 FLCTL\_PRGBRST\_DATA3\_3 Register (offset = 09Ch)

Flash Program Burst Data3 Input Register (bits  $(32 \times (x + 1) - 1)$  down to  $(32 \times x)$  for  $(x = 0, 1, 2, 3)$ )

**Figure 8-33. FLCTL\_PRGBRST\_DATA3\_3 Register**

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

**Table 8-39. FLCTL\_PRGBRST\_DATA3\_3 Register Description**

Bit	Field	Type	Reset	Description
31-0	DATAIN <sup>(1)</sup>	RW	FFFF_FF FFh	Program Burst 128 bit Data Word 3 (bits $(32 \times (x + 1) - 1)$ down to $(32 \times x)$ for $(x = 0, 1, 2, 3)$ )

<sup>(1)</sup> This bit field is writable **only** when burst status (18:16) of the FLCTL\_PRGBRST\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

### 8.4.28 FLCTL\_ERASE\_CTLSTAT Register (offset = 00A0h)

Flash Erase Control and Status Register

**Figure 8-34. FLCTL\_ERASE\_CTLSTAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												CLR_STAT	ADDR_ERR	STATUS	
r	r	r	r	r	r	r	r	r	r	r	r	w	r-0	r-0	r-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TYPE	MODE	STAR T	
r	r	r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	w

**Table 8-40. FLCTL\_ERASE\_CTLSTAT Register Description**

Bit	Field	Type	Reset	Description
31-20	Reserved	R	NA	Reserved. Reads return 0h
19	CLR_STAT <sup>(1)</sup>	W	NA	Write 1 to clear status bits 18-16 of this register Write '0' has no effect
18	ADDR_ERR	R	0h	If 1, indicates that Erase Operation was terminated due to attempted erase of reserved memory address
17-16	STATUS	R	0h	Reflects the status of erase operations in the Flash memory 00b = Idle (no program operation currently active) 01b = Erase operation triggered to START but pending 10b = Erase operation in progress 11b = Erase operation completed (status of completed erase remains in this state unless explicitly cleared by software)
15-4	Reserved	R	NA	Reserved. Reads return 0h
3-2	TYPE <sup>(2)</sup>	RW	0h	Type of memory that erase operation is carried out on (don't care if mass erase is set to 1) 00b = Main Memory 01b = Information memory 10b = Reserved 11b = Reserved
1	MODE <sup>(2)</sup>	RW	0h	Controls erase mode selected by application 0b = Sector Erase (controlled by FLTCTL_ERASE_SECTADDR) 1b = Mass Erase (includes all Main and Information memory sectors that don't have corresponding WE bits set) <sup>(3)</sup>
0	START <sup>(4)(2)</sup>	W	NA	Write 1 triggers start of Erase operation

<sup>(1)</sup> Write 1 to CLR\_STAT clears the status bits 18:16 only when erase status (17:16) shows completion state. In all other cases, write-1 has no effect. This is to allow deterministic behavior

<sup>(2)</sup> This bit field is writable **only** when status (17:16) of the FLCTL\_ERASE\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

<sup>(3)</sup> The application must ensure that the writes in this mode follow the LSB and MSB loading order within the 128-bit address boundary. See [Section 8.3.2.2](#) for more details.

<sup>(4)</sup> Writes to the START bit are ignored if the device is in Low-Frequency Active and Low-Frequency LPM0 modes of operation

### 8.4.29 FLCTL\_ERASE\_SECTADDR Register (offset = 00A4h)

Flash Erase Sector Address Register

**Figure 8-35. FLCTL\_ERASE\_SECTADDR Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										SECT_ADDRESS					
r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SECT_ADDRESS															
rw-0	rw-0	rw-0	rw-0	r	r	r	r	r	r	r	r	r	r	r	r

**Table 8-41. FLCTL\_ERASE\_SECTADDR Register Description**

Bit	Field	Type	Reset	Description
31-22	Reserved	R	0h	Reserved. Reads return 0h
21-0	SECT_ADDRESS <sup>(1)(2)</sup>	RW	0h	Address of Sector being Erased. Offset from 0h, with 0h as start address of the type of memory region selected  If memory type is set to Information/Main memory, Bits 11-0 are always 0 (forced sector boundary of 4KB)

<sup>(1)</sup> Start Address is set as max of 4MB for future enhancement purposes. To know actual amount of Flash memory available, refer to the device data sheet

<sup>(2)</sup> This bit field is writable **only** when status (17:16) of the FLCTL\_ERASE\_CTLSTAT shows the Idle state. In all other cases, the bits remain locked so as to not disrupt an operation that is in progress.

### **8.4.30 FLCTL\_BANK0\_INFO\_WEPROT Register (offset = 00B0h)**

Flash Information Memory Bank0 Write/Erase Protection Register

**Figure 8-36. FLCTL\_BANK0\_INFO\_WEPROT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														PROT 1	PROT 0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw-1 rw-1

**Table 8-42. FLCTL\_BANK0\_INFO\_WEPROT Register Description**

Bit	Field	Type	Reset	Description
31-2	Reserved	R	NA	Reserved. Reads return 0h
1	PROT <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 1 from program or erase operations
0	PROT <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 0 from program or erase operations

<sup>(1)</sup> If this Sector falls under a secure memory zone, its WEPROT bit is always set to 1, and cannot be overridden to 0.

<sup>(2)</sup> This bit field is writable **only** when status fields of the FLCTL\_PRG\_CTLSTAT, FLCTL\_PRGBRST\_CTLSTAT, and FLCTL\_ERASE\_CTLSTAT all show the Idle state. In all other cases, the bits remain locked so as to not disrupt an erase or program operation that is in progress.

### 8.4.31 FLCTL\_BANK0\_MAIN\_WEPROT Register (offset = 00B4h)

Flash Main Memory Bank0 Write/Erase Protection Register

**Figure 8-37. FLCTL\_BANK0\_MAIN\_WEPROT Register**

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

**Table 8-43. FLCTL\_BANK0\_MAIN\_WEPROT Register Description**

Bit	Field	Type	Reset	Description
31	PROT31 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 31 from program or erase operations
30	PROT30 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 30 from program or erase operations
29	PROT29 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 29 from program or erase operations
28	PROT28 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 28 from program or erase operations
27	PROT27 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 27 from program or erase operations
26	PROT26 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 26 from program or erase operations
25	PROT25 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 25 from program or erase operations
24	PROT24 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 24 from program or erase operations
23	PROT23 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 23 from program or erase operations
22	PROT22 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 22 from program or erase operations
21	PROT21 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 21 from program or erase operations
20	PROT20 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 20 from program or erase operations
19	PROT19 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 19 from program or erase operations
18	PROT18 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 18 from program or erase operations
17	PROT17 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 17 from program or erase operations
16	PROT16 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 16 from program or erase operations
15	PROT15 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 15 from program or erase operations
14	PROT14 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 14 from program or erase operations
13	PROT13 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 13 from program or erase operations
12	PROT12 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 12 from program or erase operations
11	PROT11 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 11 from program or erase operations
10	PROT10 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 10 from program or erase operations
9	PROT9 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 9 from program or erase operations
8	PROT8 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 8 from program or erase operations
7	PROT7 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 7 from program or erase operations
6	PROT6 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 6 from program or erase operations
5	PROT5 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 5 from program or erase operations
4	PROT4 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 4 from program or erase operations
3	PROT3 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 3 from program or erase operations
2	PROT2 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 2 from program or erase operations
1	PROT1 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 1 from program or erase operations
0	PROT0 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 0 from program or erase operations

<sup>(1)</sup> If this Sector falls under a secure memory zone, its WEPROT bit is always set to 1, and cannot be overridden to 0

<sup>(2)</sup> This bit field is writable **only** when status fields of the FLCTL\_PRG\_CTLSTAT, FLCTL\_PRGBRST\_CTLSTAT and the FLCTL\_ERASE\_CTLSTAT **ALL** show the Idle state. In all other cases, the bits remain locked so as to not disrupt an erase or program operation that is in progress.

---

**NOTE:** In case Bank0 of the Main Memory contains less than 32 sectors, upper bits behave as reserved. Refer to the appropriate data sheet for amount of Main Memory available on the device

---

### 8.4.32 FLCTL\_BANK1\_INFO\_WEPROT Register (offset = 00C0h)

Flash Information Memory Bank1 Write/Erase Protection Register

**Figure 8-38. FLCTL\_BANK1\_INFO\_WEPROT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw-1	rw-1

**Table 8-44. FLCTL\_BANK1\_INFO\_WEPROT Register Description**

Bit	Field	Type	Reset	Description
31-2	Reserved	R	NA	Reserved. Reads return 0h
1	PROT1 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 1 from program or erase operations
0	PROT0 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 0 from program or erase operations

<sup>(1)</sup> If this Sector falls under a secure memory zone, its WE bit is always set to 1, and cannot be overridden to 0

<sup>(2)</sup> This bit field is writable **only** when status fields of the FLCTL\_PRG\_CTLSTAT, FLCTL\_PRGBRST\_CTLSTAT and the FLCTL\_ERASE\_CTLSTAT **ALL** show the Idle state. In all other cases, the bits remain locked so as to not disrupt an erase or program operation that is in progress.

### 8.4.33 FLCTL\_BANK1\_MAIN\_WEPROT Register (offset = 00C4h)

Flash Main Memory Bank1 Write/Erase Protection Register

**Figure 8-39. FLCTL\_BANK1\_MAIN\_WEPROT Register**

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

**Table 8-45. FLCTL\_BANK1\_MAIN\_WEPROT Register Description**

Bit	Field	Type	Reset	Description
31	PROT31 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 31 from program or erase operations
30	PROT30 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 30 from program or erase operations
29	PROT29 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 29 from program or erase operations
28	PROT28 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 28 from program or erase operations
27	PROT27 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 27 from program or erase operations
26	PROT26 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 26 from program or erase operations
25	PROT25 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 25 from program or erase operations
24	PROT24 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 24 from program or erase operations
23	PROT23 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 23 from program or erase operations
22	PROT22 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 22 from program or erase operations
21	PROT21 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 21 from program or erase operations
20	PROT20 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 20 from program or erase operations
19	PROT19 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 19 from program or erase operations
18	PROT18 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 18 from program or erase operations
17	PROT17 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 17 from program or erase operations
16	PROT16 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 16 from program or erase operations
15	PROT15 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 15 from program or erase operations
14	PROT14 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 14 from program or erase operations
13	PROT13 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 13 from program or erase operations
12	PROT12 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 12 from program or erase operations
11	PROT11 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 11 from program or erase operations
10	PROT10 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 10 from program or erase operations
9	PROT9 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 9 from program or erase operations
8	PROT8 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 8 from program or erase operations
7	PROT7 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 7 from program or erase operations
6	PROT6 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 6 from program or erase operations
5	PROT5 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 5 from program or erase operations
4	PROT4 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 4 from program or erase operations
3	PROT3 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 3 from program or erase operations
2	PROT2 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 2 from program or erase operations
1	PROT1 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 1 from program or erase operations
0	PROT0 <sup>(1)(2)</sup>	RW	1h	If set to 1, protects Sector 0 from program or erase operations

<sup>(1)</sup> If this Sector falls under a secure memory zone, its WE bit is always set to 1, and cannot be overridden to 0

<sup>(2)</sup> This bit field is writable **only** when status fields of the FLCTL\_PRG\_CTLSTAT, FLCTL\_PRGBRST\_CTLSTAT and the FLCTL\_ERASE\_CTLSTAT **ALL** show the Idle state. In all other cases, the bits remain locked so as to not disrupt an erase or program operation that is in progress.

---

**NOTE:** In case Bank1 of the Main Memory contains less than 32 sectors, upper bits behave as reserved. Refer to the appropriate data sheet for the amount of Main Memory available on the device

---

#### **8.4.34 FLCTL\_BMRK\_CTLSTAT Register (offset = 00D0h)**

Flash Benchmark Control and Status Register

**Figure 8-40. FLCTL\_BMRK\_CTLSTAT Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CMP_SEL	CMP_EN	D_BMRK	I_BMRK
r	r	r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0

**Table 8-46. FLCTL\_BMRK\_CTLSTAT Register Description**

Bit	Field	Type	Reset	Description
31-4	Reserved	R	NA	Reserved. Reads return 0h
3	CMP_SEL	RW	0h	Selects which benchmark register should be compared against the threshold 0b = Compares the Instruction Benchmark Register against the threshold value 1b = Compares the Data Benchmark Register against the threshold value
2	CMP_EN	RW	0h	When 1, enables comparison of the Instruction or Data Benchmark Registers against the threshold value
1	D_BMRK	RW	0h	When 1, increments the Data Benchmark count register on each data read access to the Flash
0	I_BMRK	RW	0h	When 1, increments the Instruction Benchmark count register on each instruction fetch to the Flash

### 8.4.35 FLCTL\_BMRK\_IFETCH Register (offset = 00D4h)

Flash Benchmark Instruction Fetch Count Register

**Figure 8-41. FLCTL\_BMRK\_IFETCH Register**

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

**Table 8-47. FLCTL\_BMRK\_IFETCH Register Description**

Bit	Field	Type	Reset	Description
31-0	COUNT	RW	0h	Reflects the number of Instruction Fetches to the Flash (increments by one on each fetch)

#### **8.4.36 FLCTL\_BMRK\_DREAD Register (offset = 00D8h)**

Flash Benchmark Data Read Count Register

**Figure 8-42. FLCTL\_BMRK\_DREAD Register**

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

**Table 8-48. FLCTL\_BMRK\_DREAD Register Description**

Bit	Field	Type	Reset	Description
31-0	COUNT	RW	0h	Reflects the number of Data Read operations to the Flash (increments by one on each read)

### 8.4.37 FLCTL\_BMRK\_CMP Register (offset = 00DCh)

Flash Benchmark Count Compare Register

**Figure 8-43. FLCTL\_BMRK\_CMP Register**

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

**Table 8-49. FLCTL\_BMRK\_CMP Register Description**

Bit	Field	Type	Reset	Description
31-0	COUNT	RW	0001_00 00h	Reflects the threshold value that is compared against either the IFETCH or DREAD Benchmark Counters

### **8.4.38 FLCTL\_IFG Register (offset = 0F0h)**

Flash Interrupt Flag Register

**Figure 8-44. FLCTL\_IFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PRG_ERR	BMRK	Reserved		ERASE	PRGB	PRG	AVPST	AVPRE	RDBRST
r	r	r	r	r	r	r	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

**Table 8-50. FLCTL\_IFG Register Description**

Bit	Field	Type	Reset	Description
31- 10	Reserved	R	NA	Reserved. Reads return 0h
9	PRG_ERR	R	0h	If set to 1, indicates a word composition error in full word write mode (possible data loss due to writes crossing over to a new 128bit boundary before full word has been composed)
8	BMRK	R	0h	If set to 1, indicates that a Benchmark Compare match occurred
7-6	Reserved	R	0h	Reserved
5	ERASE	R	0h	If set to 1, indicates that the Erase operation is complete
4	PRGB	R	0h	If set to 1, indicates that the configured Burst Program operation is complete
3	PRG	R	0h	If set to 1, indicates that a word Program operation is complete
2	AVPST	R	0h	If set to 1, indicates that the post-program verify operation has failed comparison
1	AVPRE	R	0h	If set to 1, indicates that the pre-program verify operation has detected an error
0	RDBRST	R	0h	If set to 1, indicates that the Read Burst/Compare operation is complete

### 8.4.39 FLCTL\_IE Register (offset = 0F4h)

Flash Interrupt Enable Register

**Figure 8-45. FLCTL\_IE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PRG_ERR	BMRK	Reserved		ERASE	PRGB	PRG	AVPST	AVPRE	RDBRST
r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 8-51. FLCTL\_IE Register Description**

Bit	Field	Type	Reset	Description
31- 10	Reserved	R	NA	Reserved. Reads return 0h
9	PRG_ERR	RW	0h	If set to 1, enables the Controller to generate an interrupt based on the corresponding bit in the FLCTL_IFG
8	BMRK	RW	0h	If set to 1, enables the Controller to generate an interrupt based on the corresponding bit in the FLCTL_IFG
7-6	Reserved	RW	0h	Reserved
5	ERASE	RW	0h	If set to 1, enables the Controller to generate an interrupt based on the corresponding bit in the FLCTL_IFG
4	PRGB	RW	0h	If set to 1, enables the Controller to generate an interrupt based on the corresponding bit in the FLCTL_IFG
3	PRG	RW	0h	If set to 1, enables the Controller to generate an interrupt based on the corresponding bit in the FLCTL_IFG
2	AVPST	RW	0h	If set to 1, enables the Controller to generate an interrupt based on the corresponding bit in the FLCTL_IFG
1	AVPRE	RW	0h	If set to 1, enables the Controller to generate an interrupt based on the corresponding bit in the FLCTL_IFG
0	RDBRST	RW	0h	If set to 1, enables the Controller to generate an interrupt based on the corresponding bit in the FLCTL_IFG

#### **8.4.40 FLCTL\_CLRIFG Register (offset = 0F8h)**

Flash Clear Interrupt Flag Register

**Figure 8-46. FLCTL\_CLRIFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PRG_ERR	BMRK	Reserved		ERASE	PRGB	PRG	AVPST	AVPRE	RDBRST
r	r	r	r	r	r	w	w	w	w	w	w	w	w	w	w

**Table 8-52. FLCTL\_CLRIFG Register Description**

Bit	Field	Type	Reset	Description
31- 10	Reserved	R	NA	Reserved. Reads return 0h
9	PRG_ERR	W	NA	Write 1 clears the corresponding interrupt flag bit in the FLCTL_IFG
8	BMRK	W	NA	Write 1 clears the corresponding interrupt flag bit in the FLCTL_IFG
7-6	Reserved	W	NA	Reserved
5	ERASE	W	NA	Write 1 clears the corresponding interrupt flag bit in the FLCTL_IFG
4	PRGB	W	NA	Write 1 clears the corresponding interrupt flag bit in the FLCTL_IFG
3	PRG	W	NA	Write 1 clears the corresponding interrupt flag bit in the FLCTL_IFG
2	AVPST	W	NA	Write 1 clears the corresponding interrupt flag bit in the FLCTL_IFG
1	AVPRE	W	NA	Write 1 clears the corresponding interrupt flag bit in the FLCTL_IFG
0	RDBRST	W	NA	Write 1 clears the corresponding interrupt flag bit in the FLCTL_IFG

#### 8.4.41 FLCTL\_SETIFG Register (offset = 0FCh)

Flash Set Interrupt Flag Register

**Figure 8-47. FLCTL\_SETIFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PRG_ERR	BMRK	Reserved		ERASE	PRGB	PRG	AVPST	AVPRE	RDBRST
r	r	r	r	r	r	w	w	w	w	w	w	w	w	w	w

**Table 8-53. FLCTL\_SETIFG Register Description**

Bit	Field	Type	Reset	Description
31- 10	Reserved	R	NA	Reserved. Reads return 0h
9	PRG_ERR	W	NA	Write 1 sets the corresponding interrupt flag bit in the FLCTL_IFG
8	BMRK	W	NA	Write 1 sets the corresponding interrupt flag bit in the FLCTL_IFG
7-6	Reserved	W	NA	Reserved
5	ERASE	W	NA	Write 1 sets the corresponding interrupt flag bit in the FLCTL_IFG
4	PRGB	W	NA	Write 1 sets the corresponding interrupt flag bit in the FLCTL_IFG
3	PRG	W	NA	Write 1 sets the corresponding interrupt flag bit in the FLCTL_IFG
2	AVPST	W	NA	Write 1 sets the corresponding interrupt flag bit in the FLCTL_IFG
1	AVPRE	W	NA	Write 1 sets the corresponding interrupt flag bit in the FLCTL_IFG
0	RDBRST	W	NA	Write 1 sets the corresponding interrupt flag bit in the FLCTL_IFG

#### **8.4.42 FLCTL\_READ\_TIMCTL Register (offset = 0100h)**

Flash Read Timing Control Register. Applies for normal read operations.

**Figure 8-48. FLCTL\_READ\_TIMCTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SETUP_LONG							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IREF_BOOST1				Reserved				SETUP							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

**Table 8-54. FLCTL\_READ\_TIMCTL Register Description**

Bit	Field	Type	Reset	Description
31-24	Reserved	R	NA	Reserved. Always returns 0h
23-16	SETUP_LONG	R	NA	This field defines the length of the Setup time into read mode when the device is recovering from one of the following conditions Moving from standby to active state in low-frequency active mode Recovering from the LDO Boost operation after a Mass Erase
15-12	IREF_BOOST1	R	NA	Length of IREF_BOOST1 signal of the Flash memory
11-8	Reserved	R	NA	Reserved. Always returns 0h
7-0	SETUP <sup>(1)</sup>	R	NA	Length of the Setup phase for this operation

<sup>(1)</sup> All delays are in terms of clock cycles of a 5-MHz reference clock source

#### 8.4.43 FLCTL\_READMARGIN\_TIMCTL Register (offset = 0104h)

Flash Read Margin Timing Control Register. Applies for margin0/1 read operations.

**Figure 8-49. FLCTL\_READMARGIN\_TIMCTL Register**

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

**Table 8-55. FLCTL\_READMARGIN\_TIMCTL Register Description**

Bit	Field	Type	Reset	Description
31-8	Reserved	R	NA	Reserved. Always returns 0h
7-0	SETUP <sup>(1)</sup>	R	NA	Length of the Setup phase for this operation

<sup>(1)</sup> All delays are in terms of clock cycles of a 5-MHz reference clock source

#### **8.4.44 FLCTL\_PRGVER\_TIMCTL Register (offset = 0108h)**

Flash Program Verify Timing Control Register

**Figure 8-50. FLCTL\_PRGVER\_TIMCTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HOLD				ACTIVE				SETUP							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

**Table 8-56. FLCTL\_PRGVER\_TIMCTL Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	NA	Reserved. Reads return 0h
15-12	HOLD <sup>(1)</sup>	R	NA	Length of the Hold phase for this operation
11-8	ACTIVE <sup>(1)</sup>	R	NA	Length of the Active phase for this operation
7-0	SETUP <sup>(1)</sup>	R	NA	Length of the Setup phase for this operation

<sup>(1)</sup> All delays are in terms of clock cycles of a 5-MHz reference clock source

#### 8.4.45 FLCTL\_ERSVER\_TIMCTL Register (offset = 010Ch)

Flash Erase Verify Timing Control Register

**Figure 8-51. FLCTL\_ERSVER\_TIMCTL Register**

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

**Table 8-57. FLCTL\_ERSVER\_TIMCTL Register Description**

Bit	Field	Type	Reset	Description
31-8	Reserved	R	NA	Reserved. Reads return 0h
7-0	SETUP <sup>(1)</sup>	R	NA	Length of the Setup phase for this operation

<sup>(1)</sup> All delays are in terms of clock cycles of a 5-MHz reference clock source

#### **8.4.46 FLCTL\_PROGRAM\_TIMCTL Register (offset = 0114h)**

Flash Program Timing Control Register

**Figure 8-52. FLCTL\_PROGRAM\_TIMCTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HOLD				ACTIVE											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTIVE								SETUP							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

**Table 8-58. FLCTL\_PROGRAM\_TIMCTL Register Description**

Bit	Field	Type	Reset	Description
31-28	HOLD <sup>(1)</sup>	R	NA	Length of the Hold phase for this operation
27-8	ACTIVE <sup>(1)</sup>	R	NA	Length of the Active phase for this operation
7-0	SETUP <sup>(1)</sup>	R	NA	Length of the Setup phase for this operation

<sup>(1)</sup> All delays are in terms of clock cycles of a 5-MHz reference clock source

#### 8.4.47 FLCTL\_ERASE\_TIMCTL Register (offset = 0118h)

Flash Erase Timing Control Register

**Figure 8-53. FLCTL\_ERASE\_TIMCTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HOLD				ACTIVE											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTIVE								SETUP							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

**Table 8-59. FLCTL\_ERASE\_TIMCTL Register Description**

Bit	Field	Type	Reset	Description
31-28	HOLD <sup>(1)</sup>	R	NA	Length of the Hold phase for this operation
27-8	ACTIVE <sup>(1)</sup>	R	NA	Length of the Active phase for this operation
7-0	SETUP <sup>(1)</sup>	R	NA	Length of the Setup phase for this operation

<sup>(1)</sup> All delays are in terms of clock cycles of a 5-MHz reference clock source

#### **8.4.48 FLCTL\_MASSERASE\_TIMCTL Register (offset = 011Ch)**

Flash Mass Erase Timing Control Register

**Figure 8-54. FLCTL\_MASSERASE\_TIMCTL Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOST_HOLD								BOOST_ACTIVE							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

**Table 8-60. FLCTL\_MASSERASE\_TIMCTL Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	NA	Reserved. Reads return 0h
15-8	BOOST_HOLD <sup>(1)</sup>	R	NA	Length for which Flash deactivates the LDO Boost signal before processing any new commands
7-0	BOOST_ACTIVE <sup>(1)</sup>	R	NA	Length of the time for which LDO Boost Signal is kept active

<sup>(1)</sup> All delays are in terms of clock cycles of a 5-MHz reference clock source

#### 8.4.49 FLCTL\_BURSTPRG\_TIMCTL Register (offset = 0120h)

Flash Burst Program Timing Control Register. Applies for Subsequent (except first) program pulses during Burst Program Operations.

**Figure 8-55. FLCTL\_BURSTPRG\_TIMCTL Register**

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

**Table 8-61. FLCTL\_BURSTPRG\_TIMCTL Register Description**

Bit	Field	Type	Reset	Description
31-28	Reserved	R	NA	Reserved for future use.
27-8	ACTIVE <sup>(1)</sup>	R	NA	Length of the Active phase for this operation
7-0	Reserved	R	NA	Reserved for future use.

<sup>(1)</sup> All delays are in terms of clock cycles of a 5-MHz reference clock source

This chapter describes the features and use of the MSP432P4xx DMA controller module.

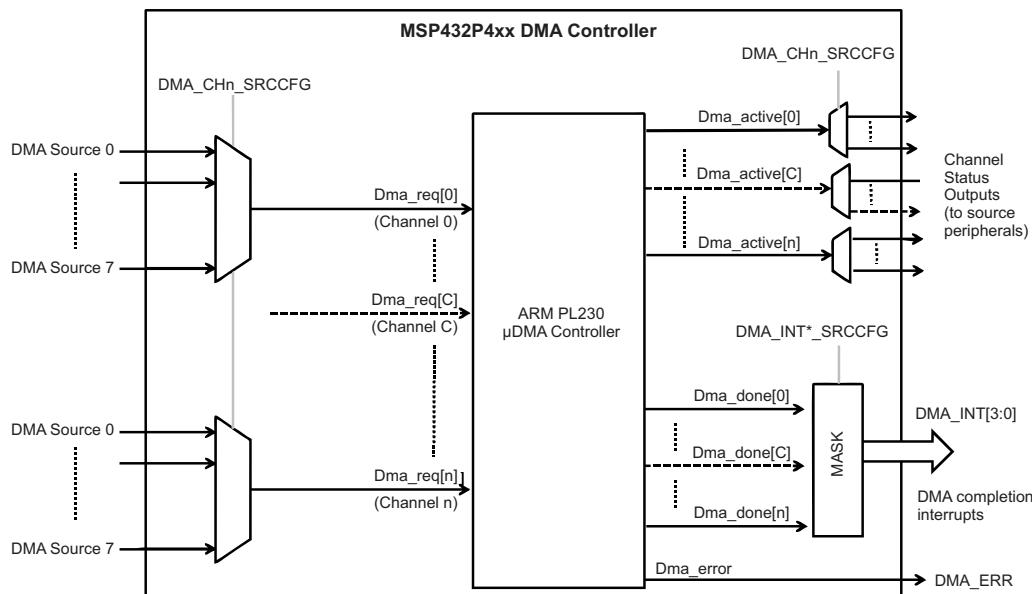
Topic	Page
9.1 DMA Introduction .....	444
9.2 DMA Operation.....	445
9.3 DMA Registers .....	469

## 9.1 DMA Introduction

MSP432P4xx DMA is built around the ARM PL230 microDMA controller ( $\mu$ DMAC) (see [Figure 9-1](#)). The  $\mu$ DMAC is an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-Chip (SoC) peripheral that is developed, tested, and licensed by ARM.

The principal features are:

- Compatible with AHB-Lite for the DMA transfers
- Compatible with APB for programming the registers
- Single AHB-Lite master for transferring data using a 32-bit address bus and 32-bit data bus
- Multiple independent DMA channels (number of DMA channels is device specific)
- Each DMA channel has dedicated handshake signals.
- Each DMA channel has a programmable priority level.
- Multiple channels with same priority level are arbitrated using a fixed priority that is determined by the DMA channel number.
- Supports multiple transfer types:
  - Memory-to-memory transfers
  - Memory-to-peripheral transfers
  - Peripheral-to-memory transfers
- Supports multiple DMA cycle types
- Supports multiple DMA transfer data widths
- Each DMA channel can access a primary and an alternate channel control data structure.
- All of the channel control data is stored in system memory in little-endian format.
- Performs all DMA transfers using the SINGLE AHB-Lite burst type
- Destination data width is equal to the source data width.
- Number of transfers in a single DMA cycle can be programmed from 1 to 1024.
- Transfer address increment can be greater than the data width.
- Single output to indicate when an ERROR condition occurs on the AHB bus
- Automatic low-power mode entry when not in use
- Triggers for each channel can be selected by the user.
- Software trigger support for each channel
- Raw and masked interrupts for optimal interrupt processing



**Figure 9-1. DMA Block Diagram**

**NOTE:** **Legend:** When [C] is used in any signal name in this chapter, it refers to channel number C. For example, dma\_done[C] means dma\_done for channel number C.

**dma\_done:** Signal generated after the DMA cycle from a channel is complete. It is used for generation of the interrupt signal.

**dma\_active:** Signal generated when a channel is serviced by the DMA controller.

**dma\_req:** Input to the DMA controller, asserted when any module or software trigger is active.

## 9.2 DMA Operation

The following sections describe the operational functionality of the controller:

- APB slave interface (see [Section 9.2.1](#))
- AHB master interface (see [Section 9.2.2](#))
- DMA control interface (see [Section 9.2.3](#))
- Channel control data structure (see [Section 9.2.4](#))

### 9.2.1 APB Slave Interface

The APB slave interface connects the controller to the APB and provides a host processor with access to the registers. The APB slave interface supports reading and writing to DMA registers using a 32-bit data bus.

### 9.2.2 AHB Master Interface

The following sections describe the features of this interface

- Transfer types
- Transfer data width
- Protection control
- Address increments

### 9.2.2.1 Transfer Types

The controller supports only SINGLE AHB-LITE transfers and has no support for any kind of BURST transfer as defined in the AMBA3 AHB-Lite protocol.

### 9.2.2.2 Transfer Data Width

The controller supports data transfer sizes of 8, 16, or 32 bits. The transfer size for the source needs to be the same as the size of the destination.

The controller always uses 32-bit data transfers when it accesses a channel control data structure.

### 9.2.2.3 Protection Control

The AHB-Lite protection control signals HPROT[3:1] indicate the following protection states:

- Cacheable
- Bufferable
- Privileged

---

**NOTE:** Although these protection options are available in the DMA, the peripherals in MSP432P4xx do not differentiate access based on any of the above qualifiers.

---

Table 9-1 lists the HPROT signal encoding.

**Table 9-1. Protection Signaling**

HPROT[3] Cacheable	HPROT[2] Bufferable	HPROT[1] Privileged	HPROT[0] Data/ Opcode	Description
–	–	–	1 <sup>(1)</sup>	Data access
–	–	0	–	User access
–	–	1	–	Privileged access
–	0	–	–	Non-bufferable
–	1	–	–	Bufferable
0	–	–	–	Non-cacheable
1	–	–	–	Cacheable

<sup>(1)</sup> The controller ties HPROT[0] HIGH, to indicate a data access.

For each DMA cycle, the source transfer and destination transfer can be configured to use different protection control settings.

### 9.2.2.4 Address Increments

Configure the address increments that the controller uses when it reads the source data or when it writes the destination data. The increments available depend on the size of data packet being transferred.

Table 9-2 lists the possible combinations.

The minimum address increment must always be equal in size to the width of the data packet. The maximum address increment that the controller permits is 32 bits.

**Table 9-2. Address Increments**

Packet Data Width (bits)	Size of Address Increment
8	byte, halfword, or word
16	halfword or word
32	word

### 9.2.3 DMA Control Interface

The DMA controller uses the DMA handshake rules defined in [Table 9-3](#) when a DMA channel is enabled and the `dma_req` for that channel is not masked.

**Table 9-3. Key Handshake Rules for the DMA Controller**

Sl. No.	Rule Description
1	When <code>dma_active[C]</code> <sup>(1)</sup> is LOW, then setting <code>dma_req[C]</code> HIGH for one or more <code>hclk</code> cycles, contiguous or non-contiguous, starts a transfer for channel C.
2	The controller only permits a single <code>dma_active[]</code> to be HIGH.
3	The controller sets <code>dma_active[C]</code> <sup>(1)</sup> HIGH, when it starts a transfer for channel C.
4	For DMA cycle types other than peripheral scatter-gather, <code>dma_active[C]</code> <sup>(1)</sup> remains HIGH until the controller completes the lesser number of transfers, that either $2^R$ or <code>n_minus_1</code> specifies. In peripheral scatter-gather mode, <code>dma_active[C]</code> <sup>(1)</sup> remains HIGH during each primary-alternate pair of DMA transfers. That is, the controller performs $2^R$ transfers using the primary data structure, and then without arbitrating, it performs the lesser number of transfers, that either $2^R$ or <code>n_minus_1</code> specifies, using the alternate data structure. After completing the latter DMA transfer, <code>dma_active[C]</code> <sup>(1)</sup> goes LOW.
5	The controller sets <code>dma_active[C]</code> LOW, for at least one <code>hclk</code> cycle, before it sets <code>dma_active[C]</code> <sup>(1)</sup> or <code>dma_active[]</code> HIGH.
6	For channels that are enabled, the controller only permits a single <code>dma_done[]</code> to be HIGH.
7	If <code>dma_req[C]</code> <sup>(1)</sup> is HIGH when <code>dma_active[C]</code> <sup>(1)</sup> is HIGH then the controller only detects a request if <code>dma_req[C]</code> <sup>(1)</sup> was low for the previous clock cycle.
8	If the <code>cycle_ctrl</code> bits for a channel are 100b, 101b, 110b, or 111b then <code>dma_done[C]</code> <sup>(1)</sup> is never set HIGH.
9	When all transfers for a channel are complete, and the <code>cycle_ctrl</code> bits enable the assertion of <code>dma_done[]</code> , then at the falling edge of <code>dma_active[]</code> , the controller sets <code>dma_done[]</code> HIGH for a duration of one <code>hclk</code> cycle.
10	For DMA cycle types other than peripheral scatter-gather, on completion of $2^R$ transfers, the controller sets the read value of the <code>chnl_useburst_set [C]</code> bit to 0, if the number of remaining transfers is less than $2^R$ . In peripheral scatter-gather mode, the controller only sets the read value of the <code>chnl_useburst_set [C]</code> <sup>(1)</sup> bit to 0, if the number of transfers remaining in the alternate data structure is less than $2^R$ .
11	When the <code>chnl_req_mask_set [C]</code> <sup>(1)</sup> bit is a 1, the controller ignores requests on <code>dma_req[C]</code> <sup>(1)</sup> .
12	For a disabled channel <sup>(2)</sup> , when <code>dma_req[C]</code> is HIGH, the controller sets <code>dma_done[C]</code> <sup>(1)</sup> HIGH. This enables the controller to alert the host processor to a request, even when the channel is disabled.
13	For a disabled channel <sup>(2)</sup> , <code>dma_active[C]</code> is always LOW.

<sup>(1)</sup> When [C] is used in any signal name in this table, it refers to channel number C. For example, `dma_req[C]` means `dma_req` for channel number C.

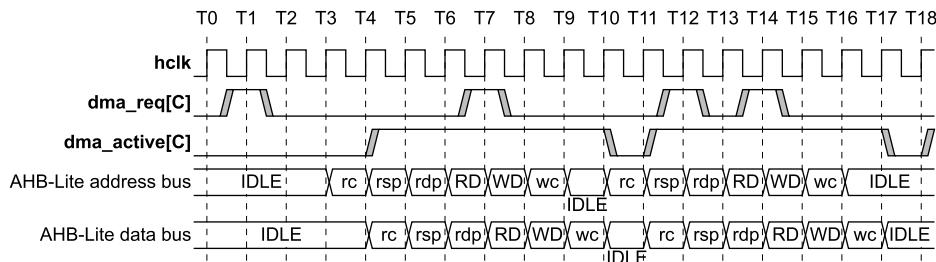
<sup>(2)</sup> Disabled channel means either the individual channel is not enabled through channel enable register or the DMA controller is not enabled using the MASTER ENABLE.

#### 9.2.3.1 DMA Signaling

The DMA signaling diagrams in this section are derived based on the handshake rules described in [Table 9-3](#).

##### 9.2.3.1.1 Signaling for Pulse Request

[Figure 9-2](#) shows the DMA request timing when a peripheral uses pulse signaling.



**Figure 9-2. DMA Signaling When Peripherals Use Pulse Requests**

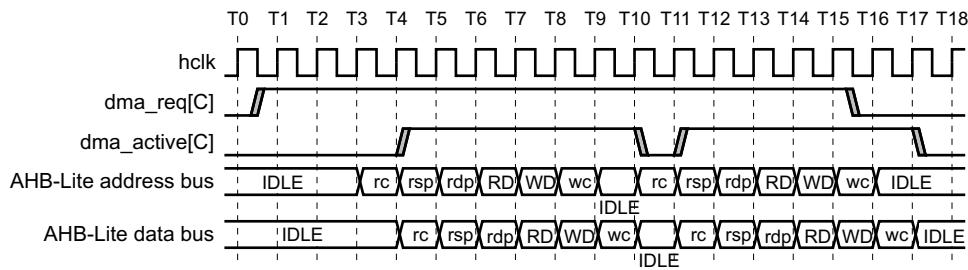
In [Figure 9-2](#):

- **T1:** The controller detects a request on channel C.

- **T4:** The controller asserts `dma_active[C]` and starts the DMA transfer for channel C.
- **T4-T7:** The controller reads the data structure, where:
  - `rc`: Reads channel configuration, `channel_cfg`
  - `rsp`: Reads source data end pointer, `src_data_end_ptr`
  - `rdp`: Reads destination data end pointer, `dst_data_end_ptr`
- **T7:** With `dma_active[C]` HIGH, the controller detects a request on channel C that was not present on the previous clock cycle. The controller includes this request during the next arbitration process.
- **T7-T9:** The controller performs the DMA transfer for channel C, where:
  - `RD`: Reads data
  - `WD`: Writes data
- **T9-T10:** The controller writes the `channel_cfg`, where:
  - `wc`: Writes channel configuration, `channel_cfg`
- **T10:** The controller deasserts `dma_active[C]` to indicate that the DMA transfer has completed.
- **T10-T11:** The controller holds `dma_active[C]` LOW for at least one `hclk` cycle.
- **T11:** If channel C is the highest priority request then the controller asserts `dma_active[C]` because of the request at T7.
- **T12:** With `dma_active[C]` HIGH, the controller detects a request on channel C that was not present on the previous clock cycle. The controller includes this request during the next arbitration process.
- **T14:** The controller ignores the request on channel C because of the pending request at T12.
- **T17:** The controller deasserts `dma_active[C]` to indicate that the DMA transfer has completed.
- **T17-T18:** The controller holds `dma_active[C]` LOW for at least one `hclk` cycle.
- **T18:** If channel C is the highest priority request then the controller asserts `dma_active[C]` because of the request at T12.

#### 9.2.3.1.2 Signaling for Level Request

Figure 9-3 shows the DMA request timing when a peripheral uses level signaling.



**Figure 9-3. DMA Signaling When Peripherals Use Level Requests**

In Figure 9-3:

- **T1:** The controller detects a request on channel C.
- **T4:** The controller asserts `dma_active[C]` and starts the DMA transfer for channel C.
- **T4-T7:** The controller reads the data structure, where:
  - `rc`: Reads channel configuration, `channel_cfg`
  - `rsp`: Reads source data end pointer, `src_data_end_ptr`
  - `rdp`: Reads destination data end pointer, `dst_data_end_ptr`
- **T7-T9:** The controller performs the DMA transfer for channel C, where:
  - `RD`: Reads data
  - `WD`: Writes data
- **T9-T10:** The controller writes the `channel_cfg`, where:
  - `wc`: Writes channel configuration, `channel_cfg`

- wc: Writes channel configuration, channel\_cfg
- **T10:** The controller deasserts dma\_active[C] to indicate that the DMA transfer has completed. The controller detects a request on channel C.
- **T10-T11:** The controller holds dma\_active[C] LOW for at least one hclk cycle.
- **T11:** If channel C is the highest priority request then the controller asserts dma\_active[C] and starts the second DMA transfer for channel C.
- **T11-T14:** The controller reads the data structure.
- **T14-T16:** The controller performs the DMA transfer for channel C.
- **T15-T16:** The peripheral acknowledges that the transfer has started and deasserts dma\_req[C].
- **T16-T17:** The controller writes the channel\_cfg.
- **T17:** The controller deasserts dma\_active[C] to indicate that the DMA transfer has completed.

### 9.2.3.2 DMA Arbitration Rate

When the controller arbitrates during a DMA transfer can be configured to reduce the latency to service a higher-priority channel.

The controller provides four bits that configure how many AHB bus transfers occur before it rearbitrates. These bits are known as the R\_power bits because 2 to the power of R determines the arbitration rate. For example, if R = 4 then the arbitration rate is 2<sup>4</sup> (that is, the controller arbitrates every 16 DMA transfers).

[Table 9-4](#) lists the arbitration rates.

**Table 9-4. AHB Bus Transfer Arbitration Interval**

R_power	Arbitrate After x DMA Transfers
0000b	x = 1
0001b	x = 2
0010b	x = 4
0011b	x = 8
0100b	x = 16
0101b	x = 32
0110b	x = 64
0111b	x = 128
1000b	x = 256
1001b	x = 512
1010b–1111b	x = 1024

---

**NOTE:** Do not assign a low-priority channel with a large R\_power, because this prevents the controller from servicing high-priority requests until it rearbitrates.

When N > 2<sup>R</sup> and is not an integer multiple of 2<sup>R</sup> then the controller always performs sequences of 2<sup>R</sup> transfers until N < 2<sup>R</sup> remain to be transferred. The controller performs the remaining N transfers at the end of the DMA cycle.

Store the value of the R\_power bits in the channel control data structure. See [Section 9.2.4.3](#) for more information about the location of the R\_power bits in the data structure.

---

### 9.2.3.3 Priority

When the controller arbitrates, it determines the next channel to service by using the following information:

- The channel number
- The priority level, default or high, that is assigned to the channel

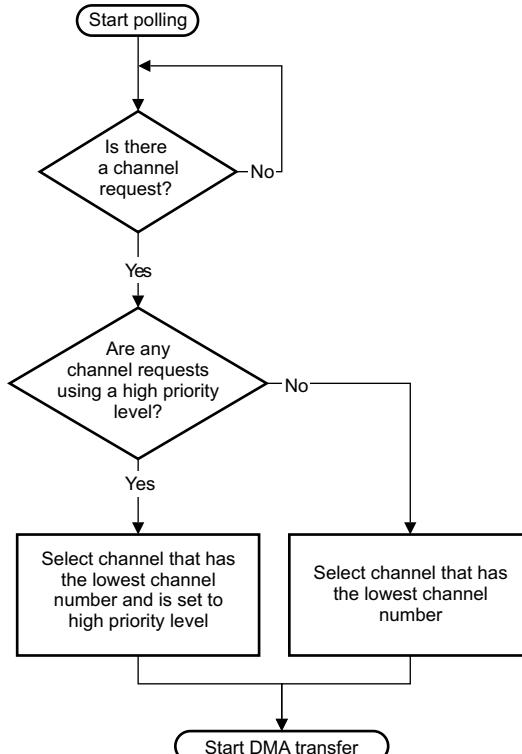
Configure each channel to use either the default priority level or a high priority level by setting the chnl\_priority\_set register (see [Section 9.3.23](#)).

Channel number zero has the highest priority and as the channel number increases, the priority of a channel decreases. [Table 9-5](#) lists the DMA channel priority levels in descending order of priority.

**Table 9-5. DMA Channel Priority**

Channel Number	Priority Level Setting	Descending Order of Channel Priority
0	High	Highest-priority DMA channel
1	High	-
2	High	-
-	High	-
-	High	-
-	High	-
30	High	-
31	High	-
0	Default	-
1	Default	-
2	Default	-
-	Default	-
-	Default	-
-	Default	-
30	Default	-
31	Default	Lowest-priority DMA channel

After a DMA transfer completes, the controller polls all the DMA channels that are available. [Figure 9-4](#) shows the process it uses to determine which DMA transfer to perform next.



**Figure 9-4. Polling Flowchart**

### 9.2.3.4 DMA Cycle Types

The cycle\_ctrl bits in the channel control data structure define how the controller performs a DMA cycle. [Table 9-6](#) lists the valid cycle\_ctrl bits.

**Table 9-6. DMA Cycle Types**

cycle_ctrl	Description
000b	Channel control data structure is invalid
001b	Basic DMA transfer
010b	Auto-request
011b	Ping-pong
100b	Memory scatter-gather using the primary data structure
101b	Memory scatter-gather using the alternate data structure
110b	Peripheral scatter-gather using the primary data structure
100b	Peripheral scatter-gather using the alternate data structure

---

**NOTE:** The cycle\_ctrl bits are located in the channel\_cfg memory location that [Section 9.2.4.3](#) describes.

---

For all cycle types, the controller arbitrates after  $2^R$  DMA transfers. If a low-priority channel has a large  $2^R$  value, then it prevents all other channels from performing a DMA transfer, until the low-priority DMA transfer completes. Therefore, make sure when setting the R\_power that the setting does not significantly increase the latency for high-priority channels.

The following sections describe different cycle types supported in DMA:

- Invalid Cycle Type (see [Section 9.2.3.4.1](#))
- Basic Cycle Type (see [Section 9.2.3.4.2](#))
- Auto-Request Cycle Type (see [Section 9.2.3.4.3](#))
- Ping-Pong Cycle Type (see [Section 9.2.3.4.4](#))
- Memory Scatter-Gather Cycle Type (see [Section 9.2.3.4.5](#))
- Peripheral Scatter-Gather Cycle Type (see [Section 9.2.3.4.6](#))

#### 9.2.3.4.1 Invalid Cycle Type

After the controller completes a DMA cycle it sets the cycle type to invalid, to prevent it from repeating the same DMA cycle.

#### 9.2.3.4.2 Basic Cycle Type

In this mode, the controller uses either the primary or alternate data structure. After the channel is enabled and the controller receives a request, then the flow for this DMA cycle is:

1. The controller performs  $2^R$  transfers. If the number of transfers remaining is zero the flow continues at step 3.
2. The controller arbitrates:
  - If a higher-priority channel is requesting service then the controller services that channel
  - If the peripheral or software signals a request to the controller on this channel, then it continues at step 1.
3. The controller sets dma\_done[C] HIGH for one hclk cycle. If the channel is enabled for interrupts, then the DMA interrupts the host processor according to the interrupt configuration.

#### 9.2.3.4.3 Auto-Request Cycle Type

When the controller operates in this mode, it is only necessary for it to receive a single request to enable it to complete the entire DMA cycle. This enables a large data transfer to occur, without significantly increasing the latency for servicing higher priority requests, or requiring multiple requests from the processor or peripheral.

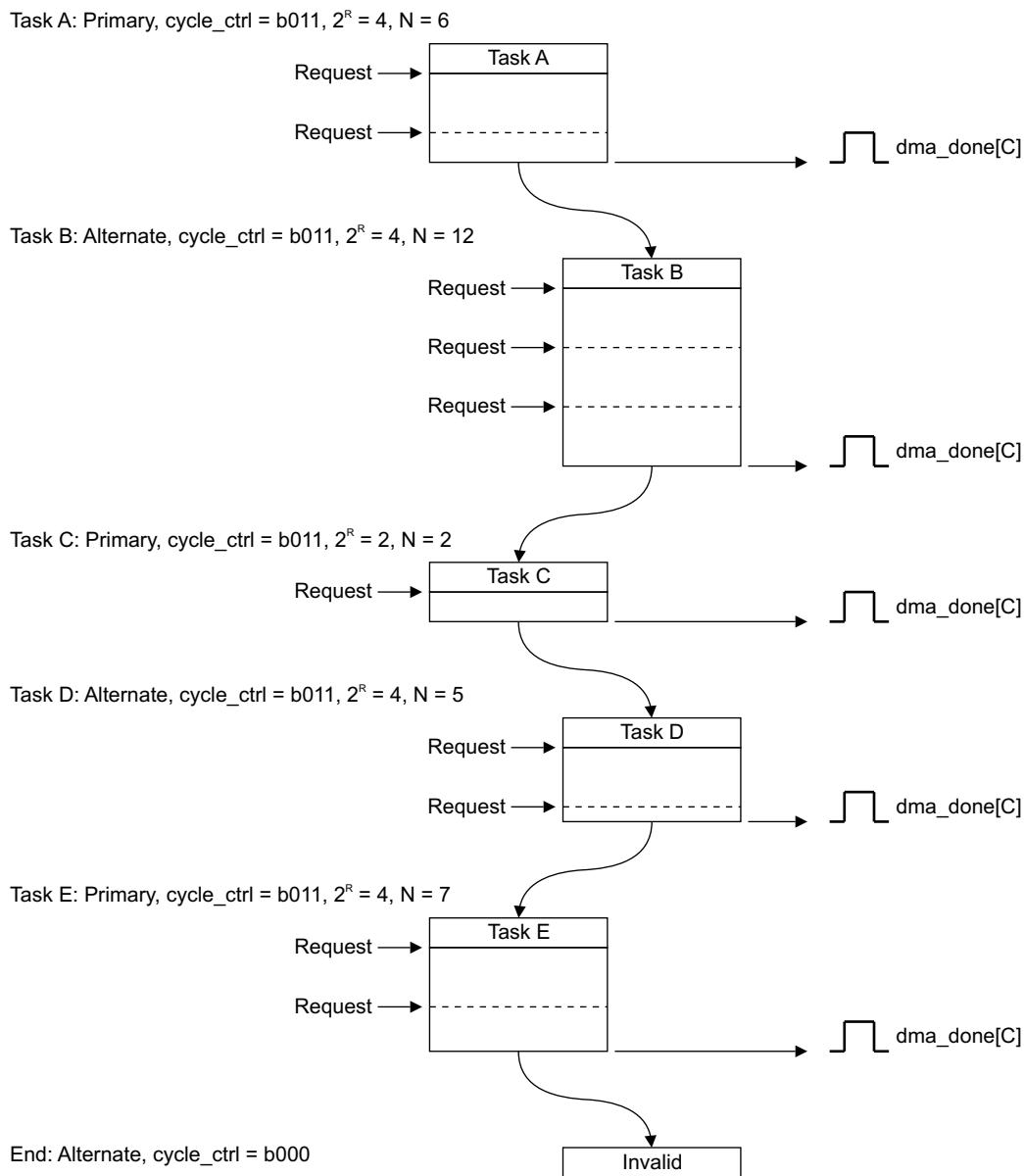
The controller can use the primary or alternate data structure. After the channel is enabled, and the controller receives a request for this channel, the flow for this DMA cycle is:

1. The controller performs  $2^R$  transfers for channel C. If the number of transfers remaining is zero the flow continues at step 3.
2. The controller arbitrates.
  - If a higher-priority channel is requesting service then the controller services that channel
  - When channel C has the highest priority, then the DMA cycle continues at step 1.
3. The controller sets `dma_done[C]` HIGH for one hclk cycle. If the channel is enabled for Interrupts, then the DMA interrupts the host processor according to the interrupt configuration.

#### 9.2.3.4.4 Ping-Pong Cycle Type

In ping-pong mode, the controller performs a DMA cycle using one of the data structures and it then performs a DMA cycle using the other data structure. The controller continues to switch from primary to alternate to primary... until it reads a data structure that is invalid, or until the host processor disables the channel.

Figure 9-5 shows an example of a ping-pong DMA transaction.



**Figure 9-5. Ping-Pong Example**

In [Figure 9-5](#):

### Task A

1. The host processor configures the primary data structure for task A.
2. The host processor configures the alternate data structure for task. This enables the controller to immediately switch to task B after task A completes, provided that a higher priority channel does not require servicing.
3. The controller receives a request and performs four DMA transfers.
4. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
5. The controller performs the remaining two DMA transfers.
6. The controller sets **dma\_done[C]** HIGH for one **hclk** cycle and enters the arbitration process. If the channel is enabled for Interrupts, then the DMA interrupts the host processor according to the interrupt configuration.

After task A completes, the host processor can configure the primary data structure for task C. This enables the controller to immediately switch to task C after task B completes, provided that a higher priority channel does not require servicing.

After the controller receives a new request for the channel and it has the highest priority then task B commences:

### Task B

1. The controller performs four DMA transfers.
2. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
3. The controller performs four DMA transfers.
4. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
5. The controller performs the remaining four DMA transfers.
6. The controller sets **dma\_done[C]** HIGH for one **hclk** cycle and enters the arbitration process. If the channel is enabled for Interrupts, then the DMA interrupts the host processor according to the interrupt configuration.

After task B completes, the host processor can configure the alternate data structure for task D.

After the controller receives a new request for the channel and it has the highest priority then task C commences (task C can be the same task as task A if no change in data structure was done):

### Task C

1. The controller performs two DMA transfers.
2. The controller sets **dma\_done[C]** HIGH for one **hclk** cycle and enters the arbitration process. If the channel is enabled for Interrupts, then the DMA interrupts the host processor according to the interrupt configuration.

After task C completes, the host processor can configure the primary data structure for task E.

After the controller receives a new request for the channel and it has the highest priority then task D commences:

### Task D

1. The controller performs four DMA transfers.
2. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
3. The controller performs the remaining DMA transfer.
4. The controller sets **dma\_done[C]** HIGH for one **hclk** cycle and enters the arbitration process. If the channel is enabled for Interrupts, then the DMA interrupts the host processor according to the interrupt configuration.

After the controller receives a new request for the channel and it has the highest priority then task E commences:

### Task E

1. The controller performs four DMA transfers.
2. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
3. The controller performs the remaining three DMA transfers.
4. The controller sets **dma\_done[C]** HIGH for one **hclk** cycle and enters the arbitration process. If the channel is enabled for Interrupts, then the DMA interrupts the host processor according to the interrupt configuration.

If the controller receives a new request for the channel and it has the highest priority, then the controller attempts to start the next task. However, because the host processor has not configured the alternate data structure, and on completion of task D the controller set the **cycle\_ctrl** bits to 000b, then the ping-pong DMA transaction completes.

---

**NOTE:** Terminate the ping-pong DMA cycle in [Figure 9-5](#) by configuring task E to be a basic DMA cycle by setting the `cycle_ctrl` field to 001b.

---

#### 9.2.3.4.4.1 Example Application Use Case for Ping-Pong Cycle

Ping-pong mode is preferred when data is generated at high speed (for example, the ADC with fast sampling rate) and the DMA should copy the data while the CPU is still processing the earlier block of data.

In many applications, the ADC output data is processed by CPU in blocks. Consider a scenario when DMA has copied a block data into the memory and interrupted CPU to process the data. The CPU started processing the data, but meanwhile the ADC is ready with another conversion and triggers the DMA. The DMA cannot copy to the earlier destination address, because the CPU has not finished processing the previous data. Ping-pong mode allows the DMA to copy the data to a new location as defined by the alternate data structure. So, while the CPU is busy processing the data copied the using primary data structure, the DMA starts filling a new block using the alternate data structure. When the CPU processes the data from alternate data structure, the DMA starts to fill the memory based on the primary data structure. By using ping-pong mode, the application can prevent loss of any data for high-data-rate requirements.

#### 9.2.3.4.5 Memory Scatter-Gather Cycle Type

In memory scatter-gather mode, the controller receives an initial request and then performs four DMA transfers using the primary data structure. After this transfer completes, the controller starts a DMA cycle using the alternate data structure. After this cycle completes, the controller performs another four DMA transfers using the primary data structure. The controller continues to switch from primary to alternate to primary until either:

- The host processor configures the alternate data structure for a basic cycle.
- The controller reads an invalid data structure.

---

**NOTE:** After the controller completes the N primary transfers, it invalidates the primary data structure by setting the `cycle_ctrl` field to 000b.

---

The controller asserts `dma_done[C]` only when the scatter-gather transaction completes using a basic cycle.

In scatter-gather mode, the controller uses the primary data structure to program the alternate data structure. [Table 9-7](#) lists the fields of the `channel_cfg` memory location for the primary data structure that must be programmed with constant values and those that can be user defined.

**Table 9-7. `channel_cfg` for a Primary Data Structure, In Memory Scatter-Gather Mode**

Bit	Field	Value	Description
<b>Constant-value fields:</b>			
[31:30]	dst_inc	10b	Configures the controller to use word increments for the address
[29:28]	dst_size	10b	Configures the controller to use word transfers
[27:26]	src_inc	10b	Configures the controller to use word increments for the address
[25:24]	src_size	10b	Configures the controller to use word transfers
[17:14]	R_power	0010b	Configures the controller to perform four DMA transfers
[3]	next_useburst	0	For a memory scatter-gather DMA cycle, this bit must be set to zero
[2:0]	cycle_ctrl	100b	Configures the controller to perform a memory scatter-gather DMA cycle
<b>User-defined values:</b>			
[23:21]	dst_prot_ctrl	-	Configures the state of HPROT when the controller writes the destination data
[20:18]	src_prot_ctrl	-	Configures the state of HPROT when the controller reads the source data

**Table 9-7. channel\_cfg for a Primary Data Structure, In Memory Scatter-Gather Mode (continued)**

Bit	Field	Value	Description
[13:4]	n_minus_1	N <sup>(1)</sup>	Configures the controller to perform N DMA transfers, where N is a multiple of four

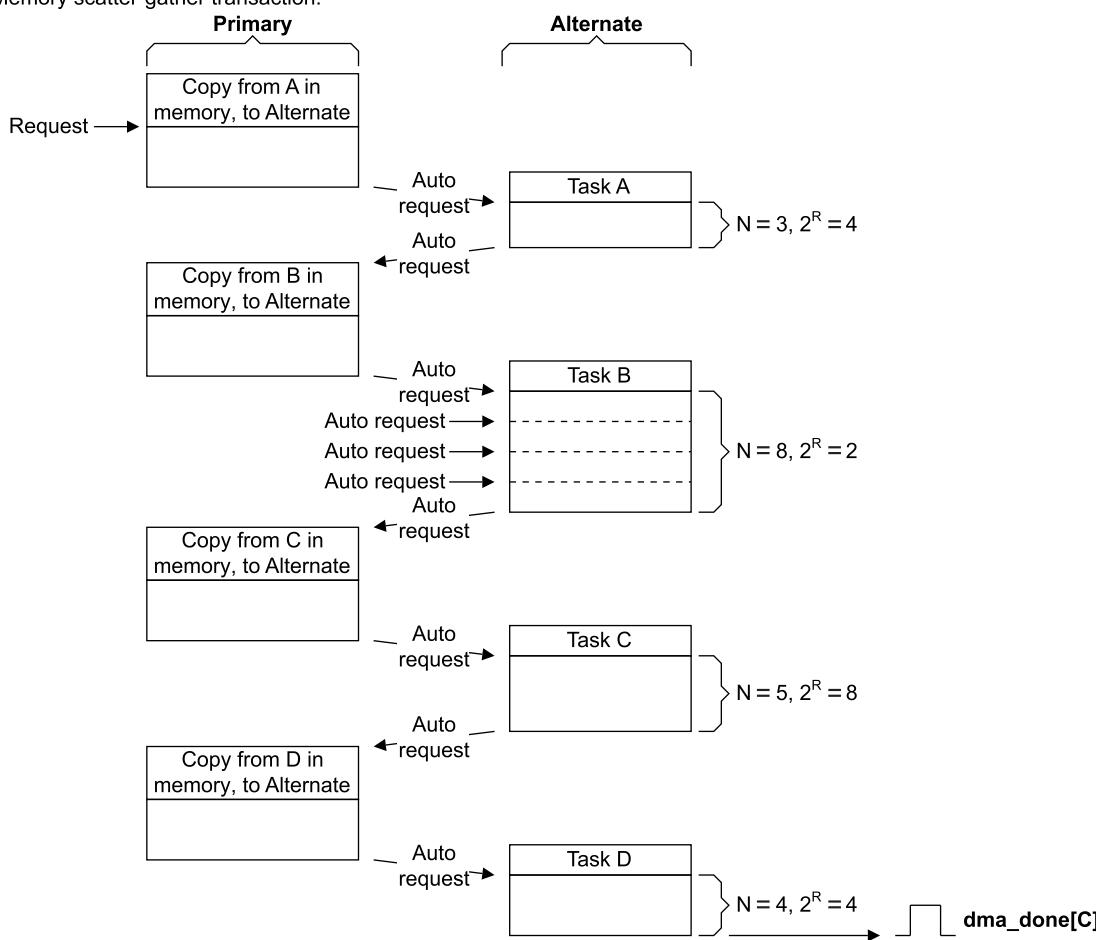
<sup>(1)</sup> Because the R\_power field is set to four, set N to be a multiple of four. The value given by N/4 is the number of times that the alternate data structure must be configured.

Figure 9-6 shows a memory scatter-gather example.

- Initialization:
1. Configure primary to enable the copy A, B, C, and D operations: cycle\_ctrl = b100,  $2^R = 4$ , N = 16.
  2. Write the primary source data to memory, using the structure shown in the following table.

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
Data for Task A	0x0A000000	0x0AE00000	cycle_ctrl b101, $2^R = 4$ , N = 3	0xFFFFFFFFXX
Data for Task B	0x0B000000	0x0BE00000	cycle_ctrl b101, $2^R = 2$ , N = 8	0xFFFFFFFFXX
Data for Task C	0x0C000000	0x0CE00000	cycle_ctrl b101, $2^R = 8$ , N = 5	0xFFFFFFFFXX
Data for Task D	0x0D000000	0x0DE00000	cycle_ctrl b001, $2^R = 4$ , N = 4	0xFFFFFFFFXX

Memory scatter-gather transaction:

**Figure 9-6. Memory Scatter-Gather Example**

In Figure 9-6:

### Initialization

1. The host processor configures the primary data structure to operate in memory scatter-gather mode by setting cycle\_ctrl to 100b. Because a data structure for a single channel consists of four words,  $2^R$  must be set to 4. In this example, there are four tasks, and therefore N is set to 16.

2. The host processor writes the data structure for tasks A, B, C, and D to the memory locations that the primary `src_data_end_ptr` specifies.
3. The host processor enables the channel.

The memory scatter-gather transaction commences when the controller receives a trigger from the configured peripheral or a software trigger from the host processor. The transaction continues as follows:

#### **Primary, copy A**

1. After receiving a request, the controller performs four DMA transfers. These transfers write the alternate data structure for task A.
2. The controller generates an auto-request for the channel and then arbitrates.

#### **Task A**

1. The controller performs task A. After it completes the task, it generates an auto-request for the channel and then arbitrates.

#### **Primary, copy B**

1. The controller performs four DMA transfers. These transfers write the alternate data structure for task B.
2. The controller generates an auto-request for the channel and then arbitrates.

#### **Task B**

1. The controller performs task B. After it completes the task, it generates an auto-request for the channel and then arbitrates.

#### **Primary, copy C**

1. The controller performs four DMA transfers. These transfers write the alternate data structure for task C.
2. The controller generates an auto-request for the channel and then arbitrates.

#### **Task C**

1. The controller performs task C. After it completes the task, it generates an auto-request for the channel and then arbitrates.

#### **Primary, copy D**

1. The controller performs four DMA transfers. These transfers write the alternate data structure for task D.
2. The controller sets the `cycle_ctrl` bits of the primary data structure to 000b, to indicate that this data structure is now invalid.
3. The controller generates an auto-request for the channel and then arbitrates.

#### **Task D**

1. The controller performs task D using a basic cycle.
2. The controller sets `dma_done[C]` HIGH for one `hclk` cycle and enters the arbitration process. If the channel is enabled for Interrupts, then the DMA interrupts the host processor according to the interrupt configuration.

#### **9.2.3.4.5.1 Example Application use case for Memory Scatter-Gather Cycle**

Memory Scatter-gather mode is useful when the user wants to accomplish multiple data transfers without requiring to configure DMA multiple times. Examples could be when the user wants to copy non-contiguous blocks of data from flash memory to SRAM or from one location of SRAM to another location in SRAM. In this case, each of the contiguous blocks of data in memory could be a separate task as in the above example.

#### **9.2.3.4.6 Peripheral Scatter-Gather Cycle Type**

In peripheral scatter-gather mode the controller receives an initial request from a peripheral and then it performs four DMA transfers using the primary data structure. It then immediately starts a DMA cycle using the alternate data structure, without rearbitrating or `dma_active[C]` going LOW.

---

**NOTE:** These are the only circumstances where the controller does not enter the arbitration process after completing a transfer using the primary data structure.

---

After this cycle completes, the controller re-arbitrates and if the controller receives a request from the peripheral that has the highest priority then it performs another four DMA transfers using the primary data structure. It then immediately starts a DMA cycle using the alternate data structure, without re-arbitrating or **dma\_active[C]** going LOW.

The controller continues to switch from primary to alternate to primary until either:

- the host processor configures the alternate data structure for a basic cycle, or
  - it reads an invalid data structure.
- 

**NOTE:** After the controller completes the N primary transfers it invalidates the primary data structure by setting the **cycle\_ctrl** field to 000b.

---

The controller asserts **dma\_done[C]** when the scatter-gather transaction completes using a basic cycle..

In scatter-gather mode, the controller uses the primary data structure to program the alternate data structure. [Table 9-8](#) lists the fields of the **channel\_cfg** memory location for the primary data structure that must be programmed with constant values and those that can be user defined.

---

**NOTE:** Peripheral scatter gather mode is almost similar to the memory scatter gather mode with the difference being that during the transfer using the alternate data structure, every time the controller arbitrates, it expects a new trigger from the peripheral to continue with the remaining transfers.

---

**Table 9-8. channel\_cfg for a Primary Data Structure, in Peripheral Scatter-Gather Mode**

Bit	Field	Value	Description
<b>Constant-value fields:</b>			
[31:30]	dst_inc	10b	Configures the controller to use word increments for the address
[29:28]	dst_size	10b	Configures the controller to use word transfers
[27:26]	src_inc	10b	Configures the controller to use word increments for the address
[25:24]	src_size	10b	Configures the controller to use word transfers
[17:14]	R_power	0010b	Configures the controller to perform four DMA transfers
[2:0]	cycle_ctrl	110b	Configures the controller to perform a peripheral scatter-gather DMA cycle
<b>User-defined values:</b>			
[23:21]	dst_prot_ctrl	-	Configures the state of <b>HPROT</b> when the controller writes the destination data
[20:18]	src_prot_ctrl	-	Configures the state of <b>HPROT</b> when the controller reads the source data
[13:4]	n_minus_1	N <sup>(1)</sup>	Configures the controller to perform N DMA transfers, where N is a multiple of four
[3]	next_useburst	-	When set to 1, the controller sets the <b>chnl_useburst_set [C]</b> bit to 1 after the alternate transfer completes

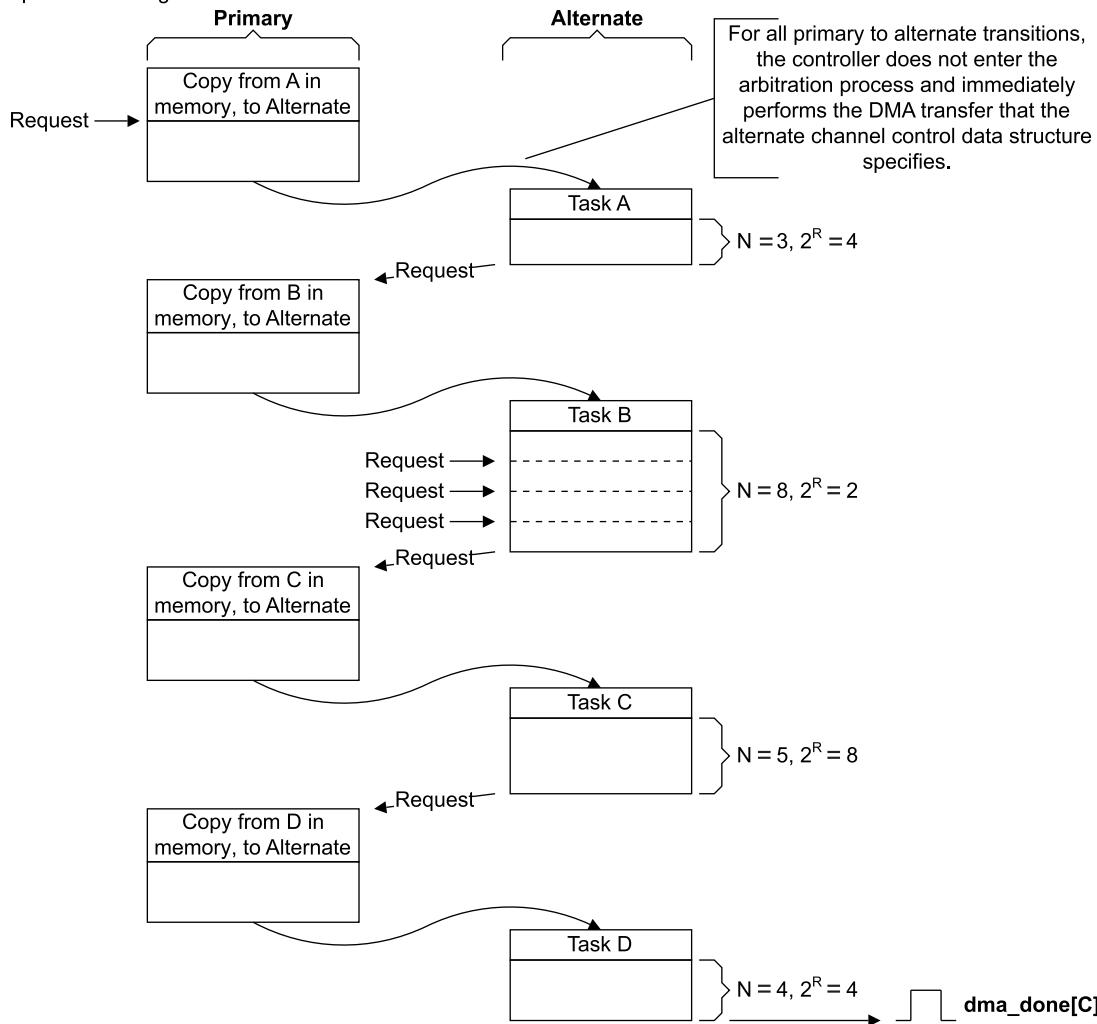
<sup>(1)</sup> Because the R\_power field is set to four, set N to be a multiple of four. The value given by N/4 is the number of times that the alternate data structure must be configured.

**Figure 9-7** shows a peripheral scatter-gather example.

- Initialization:
1. Configure primary to enable the copy A, B, C, and D operations:  $\text{cycle\_ctrl} = b110, 2^R = 4, N = 16$ .
  2. Write the primary source data in memory, using the structure shown in the following table.

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
Data for Task A	0x0A000000	0x0AE00000	cycle_ctrl = b111, $2^R = 4, N = 3$	0XXXXXXXXXX
Data for Task B	0x0B000000	0x0BE00000	cycle_ctrl = b111, $2^R = 2, N = 8$	0XXXXXXXXXX
Data for Task C	0x0C000000	0x0CE00000	cycle_ctrl = b111, $2^R = 8, N = 5$	0XXXXXXXXXX
Data for Task D	0x0D000000	0x0DE00000	cycle_ctrl = b001, $2^R = 4, N = 4$	0XXXXXXXXXX

Peripheral scatter-gather transaction:



**Figure 9-7. Peripheral Scatter-Gather Example**

In [Figure 9-7](#):

#### Initialization

1. The host processor configures the primary data structure to operate in peripheral scatter-gather mode by setting  $\text{cycle\_ctrl}$  to 110b. Because a data structure for a single channel consists of four words,  $2^R$  must be set to 4. In this example, there are four tasks, and therefore N is set to 16.
2. The host processor writes the data structure for tasks A, B, C, and D to the memory locations that the primary  $\text{src\_data\_end\_ptr}$  specifies.
3. The host processor enables the channel.

The peripheral scatter-gather transaction commences when the controller receives a trigger on associated channel. The transaction continues as follows:

#### **Primary, copy A**

1. After receiving a request, the controller performs four DMA transfers. These transfers write the alternate data structure for task A.

#### **Task A**

1. The controller performs task A.
2. After the controller completes the task it enters the arbitration process.

After the peripheral issues a new request and it has the highest priority then the process continues with:

#### **Primary, copy B**

1. The controller performs four DMA transfers. These transfers write the alternate data structure for task B.

#### **Task B**

1. The controller performs task B. To enable the controller to complete the task, the peripheral must issue a further three requests.
2. After the controller completes the task it enters the arbitration process.

After the peripheral issues a new request and it has the highest priority then the process continues with:

#### **Primary, copy C**

1. The controller performs four DMA transfers. These transfers write the alternate data structure for task C.

#### **Task C**

1. The controller performs task C.
2. After the controller completes the task it enters the arbitration process.

After the peripheral issues a new request and it has the highest priority then the process continues with:

#### **Primary, copy D**

1. The controller performs four DMA transfers. These transfers write the alternate data structure for task D.
2. The controller sets the cycle\_ctrl bits of the primary data structure to 000b, to indicate that this data structure is now invalid.

#### **Task D**

1. The controller performs task D using a basic cycle.
2. The controller sets **dma\_done[C]** HIGH for one **hclk** cycle and enters the arbitration process. If the channel is enabled for Interrupts, then the DMA interrupts the host processor according to the interrupt configuration.

#### **9.2.3.4.7 Error Signaling**

If the controller detects an ERROR response on the AHB-Lite master interface, it:

- disables the channel that corresponds to the ERROR
- sets **dma\_err** HIGH.

Typically, bus errors arise when accessing an invalid address or accessing a 16-bit peripheral using 32-bit accesses. **dma\_err** maps to a dedicated interrupt line in MSP432P4xx devices. (See [Section 9.2.6](#) for more details). After the host processor receives an interrupt due to error, it must check which channel was active when the ERROR occurred. It can do this by:

1. Reading the **chnl\_enable\_set** register to create a list of disabled channels. When a channel asserts **dma\_done[]** then the controller disables the channel. The program running on the host processor must always keep a record of which channels have recently asserted their **dma\_done[]** outputs.
2. It must compare the disabled channels list from step 1, with the record of the channels that have

recently set their `dma_done[]` outputs. The channel with no record of `dma_done[C]` being set is the channel that the ERROR occurred on.

#### 9.2.4 Channel Control Data Structure

The application must provide an area of system memory to contain the channel control data structure. This system memory must:

- Provide a contiguous area of system memory that the controller and host processor can access
- Have a base address that is an integer multiple of the total size of the channel control data structure

[Figure 9-8](#) shows the memory that the controller requires for the channel control data structure, when it uses all 32 channels and the optional alternate data structure.

Alternate data structure	Primary data structure										
Alternate_Ch_31	Primary_Ch_31										
0x3F0	0x1F0										
Alternate_Ch_30	Primary_Ch_30										
0x3E0	0x1E0										
Alternate_Ch_29	Primary_Ch_29										
0x3D0	0x1D0										
Alternate_Ch_28	Primary_Ch_28										
0x3C0	0x1C0										
Alternate_Ch_27	Primary_Ch_27										
0x3B0	0x1B0										
Alternate_Ch_26	Primary_Ch_26										
0x3A0	0x1A0										
Alternate_Ch_25	Primary_Ch_25										
0x390	0x190										
Alternate_Ch_24	Primary_Ch_24										
0x380	0x180										
Alternate_Ch_23	Primary_Ch_23										
0x370	0x170										
Alternate_Ch_22	Primary_Ch_22										
0x360	0x160										
Alternate_Ch_21	Primary_Ch_21										
0x350	0x150										
Alternate_Ch_20	Primary_Ch_20										
0x340	0x140										
Alternate_Ch_19	Primary_Ch_19										
0x330	0x130										
Alternate_Ch_18	Primary_Ch_18										
0x320	0x120										
Alternate_Ch_17	Primary_Ch_17										
0x310	0x110										
Alternate_Ch_16	Primary_Ch_16										
0x300	0x100										
Alternate_Ch_15	Primary_Ch_15										
0x2F0	0x0F0										
Alternate_Ch_14	Primary_Ch_14										
0x2E0	0x0E0										
Alternate_Ch_13	Primary_Ch_13										
0x2D0	0x0D0										
Alternate_Ch_12	Primary_Ch_12										
0x2C0	0x0C0										
Alternate_Ch_11	Primary_Ch_11										
0x2B0	0x0B0										
Alternate_Ch_10	Primary_Ch_10										
0x2A0	0x0A0										
Alternate_Ch_9	Primary_Ch_9										
0x290	0x090										
Alternate_Ch_8	Primary_Ch_8										
0x280	0x080										
Alternate_Ch_7	Primary_Ch_7										
0x270	0x070										
Alternate_Ch_6	Primary_Ch_6										
0x260	0x060										
Alternate_Ch_5	Primary_Ch_5										
0x250	0x050										
Alternate_Ch_4	Primary_Ch_4										
0x240	0x040										
Alternate_Ch_3	Primary_Ch_3										
0x230	0x030										
Alternate_Ch_2	Primary_Ch_2										
0x220	0x020										
Alternate_Ch_1	Primary_Ch_1										
0x210	0x010										
Alternate_Ch_0	Primary_Ch_0										
0x200	0x000										
	<table border="1"> <tr><td>Unused</td><td>0x00C</td></tr> <tr><td>Control</td><td>0x008</td></tr> <tr><td>Destination End Pointer</td><td>0x004</td></tr> <tr><td>Source End Pointer</td><td>0x000</td></tr> <tr><td></td><td></td></tr> </table>	Unused	0x00C	Control	0x008	Destination End Pointer	0x004	Source End Pointer	0x000		
Unused	0x00C										
Control	0x008										
Destination End Pointer	0x004										
Source End Pointer	0x000										

**Figure 9-8. Memory Map for 32 Channels, Including the Alternate Data Structure**

The example structure in [Figure 9-8](#) uses 1KB of system memory. In this example, the controller uses the lower 10 address bits to enable it to access all of the elements in the structure and therefore the base address must be at 0XXXXXX000, 0XXXXXX400, 0XXXXXX800, or 0XXXXXXC00.

Configure the base address for the primary data structure by writing the appropriate value in the `ctrl_base_ptr` register.

The amount of system memory required depends on:

- The number of DMA channels that the controller uses
- If a DMA channel uses the alternate data structure

**Table 9-9** lists the address bits that the controller uses when it accesses the elements of the channel control data structure, depending on the number of channels that the controller contains.

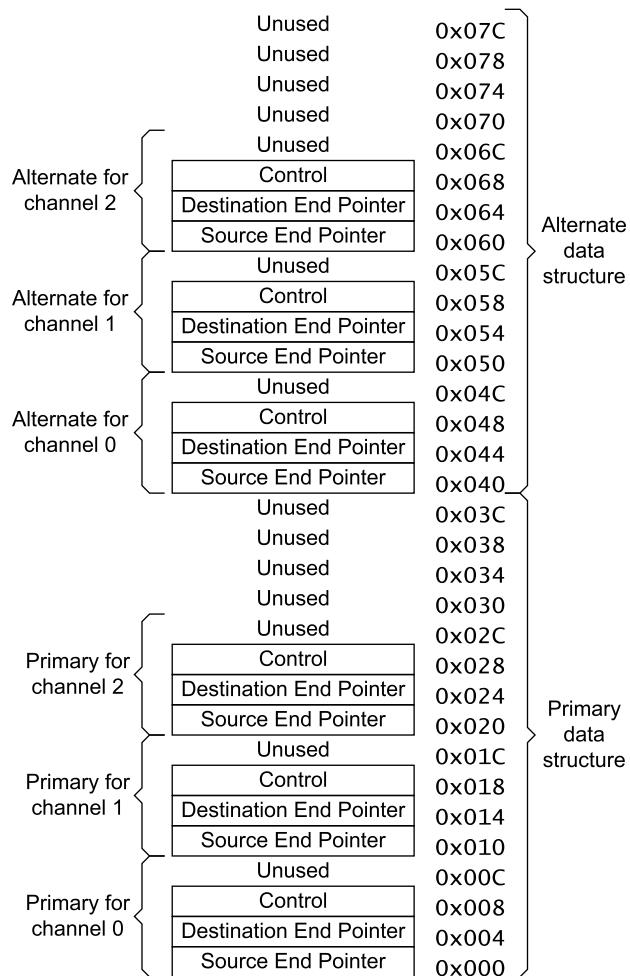
**Table 9-9. Address Bit Settings for the Channel Control Data Structure**

Number of DMA Channels Implemented	Address Bits							
	[9]	[8]	[7]	[6]	[5]	[4]	[3:0]	
1						A	0x0, 0x4, or 0x8	
2					A	C[0]		
3-4				A	C[1]	C[0]		
5-8			A	C[2]	C[1]	C[0]		
9-16		A	C[3]	C[2]	C[1]	C[0]		
17-32	A	C[4]	C[3]	C[2]	C[1]	C[0]		

Where:

- A      Selects one of the channel control data structures:
  - A = 0    Selects the primary data structure.
  - A = 1    Selects the alternate data structure.
- C[x:0]    Selects the DMA channel.
- Address[3:0]    Selects one of the control elements:
  - 0x0    Selects the source data end pointer.
  - 0x4    Selects the destination data end pointer.
  - 0x8    Selects the control data configuration.
  - 0xC    The controller does not access this address location. If required, the host processor can use this memory location as system memory.

[Figure 9-9](#) shows an example implementation where the controller uses three DMA channels and the alternate data structure.



**Figure 9-9. Memory Map for Three DMA Channels, Including the Alternate Data Structure**

The example structure in [Figure 9-9](#) uses 128 bytes of system memory. In this example, the controller uses the lower six address bits to enable it to access all of the elements in the structure and therefore the base address must be at 0XXXXXXXX00 or 0XXXXXXXX80.

[Table 9-10](#) lists the permitted base address values for the primary data structure, depending on the number of channels that the controller contains.

**Table 9-10. Permitted Base Addresses**

Number of DMA Channels	Permitted Base Addresses <sup>(1)</sup> for the Primary Data Structure
1	0XXXXXXXX00, 0XXXXXXXX20, 0XXXXXXXX40, 0XXXXXXXX60, 0XXXXXXXX80, 0XXXXXXXXA0, 0XXXXXXXXC0, 0XXXXXXXXE0
2	0XXXXXXXX00, 0XXXXXXXX40, 0XXXXXXXX80, 0XXXXXXXXC0
3-4	0XXXXXXXX00, 0XXXXXXXX80
5-8	0XXXXXXXX000, 0XXXXXX100, 0XXXXXX200, 0XXXXXX300, 0XXXXXX400, 0XXXXXX500, 0XXXXXX600, 0XXXXXX700, 0XXXXXX800, 0XXXXXX900, 0XXXXXA00, 0XXXXXB00, 0XXXXXC00, 0XXXXXD00, 0XXXXXE00, 0XXXXXF00
9-16	0XXXXXXXX000, 0XXXXXX200, 0XXXXXX400, 0XXXXXX600, 0XXXXXX800, 0XXXXXA00, 0XXXXXC00, 0XXXXXE00

<sup>(1)</sup> Where X is a hexadecimal.

**Table 9-10. Permitted Base Addresses (continued)**

Number of DMA Channels	Permitted Base Addresses <sup>(1)</sup> for the Primary Data Structure
17-32	0XXXXX000, 0XXXXX400, 0XXXXX800, 0XXXXXC00

The controller uses the system memory to enable it to access two pointers and the control information that it requires for each channel. The following subsections describe these 32-bit memory locations and how the controller calculates the DMA transfer address:

- Source data end pointer (see [Section 9.2.4.1](#))
- Destination data end pointer (see [Section 9.2.4.2](#))
- Control data configuration (see [Section 9.2.4.3](#))
- Address calculation (see [Section 9.2.4.4](#))

#### 9.2.4.1 Source Data End Pointer

The `src_data_end_ptr` memory location contains a pointer to the end address of the source data. [Table 9-11](#) lists the bit assignments for this memory location.

**Table 9-11. rc\_data\_end\_ptr Bit Assignments**

Bit	Name	Description
[31:0]	<code>src_data_end_ptr</code>	Pointer to the end address of the source data

Before the controller can perform a DMA transfer, program this memory location with the end address of the source data. The controller reads this memory location when it starts a  $2^R$  DMA transfer.

#### 9.2.4.2 Destination Data End Pointer

The `dst_data_end_ptr` memory location contains a pointer to the end address of the destination data. [Table 9-12](#) lists the bit assignments for this memory location.

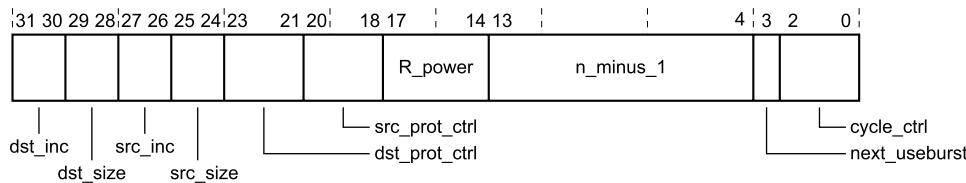
**Table 9-12. dst\_data\_end\_ptr Bit Assignments**

Bit	Name	Description
[31:0]	<code>dst_data_end_ptr</code>	Pointer to the end address of the destination data

Before the controller can perform a DMA transfer, program this memory location with the end address of the destination data. The controller reads this memory location when it starts a  $2^R$  DMA transfer.

#### 9.2.4.3 Control Data Configuration

For each DMA transfer, the `channel_cfg` memory location provides the control information for the controller. [Section 9.2.4.3](#) shows the bit assignments for this memory location.

**Figure 9-10. channel\_cfg Bit Assignments**

[Table 9-13](#) lists the bit assignments for this memory location.

**Table 9-13. channel\_cfg Bit Assignments**

Bit	Name	Description
[31:30]	dst_inc	<p>Destination address increment. The address increment depends on the source data width as follows:</p> <p>Source data width = byte</p> <ul style="list-style-type: none"> <li>00b = byte</li> <li>01b = halfword</li> <li>10b = word</li> <li>11b = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains.</li> </ul> <p>Source data width = halfword</p> <ul style="list-style-type: none"> <li>00b = reserved</li> <li>01b = halfword</li> <li>10b = word</li> <li>11b = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains.</li> </ul> <p>Source data width = word</p> <ul style="list-style-type: none"> <li>00b = reserved</li> <li>01b = reserved</li> <li>10b = word</li> <li>11b = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains.</li> </ul>
[29:28]	dst_size	<p>Destination data size</p> <p>NOTE: Set dst_size equal to src_size.</p>
[27:26]	src_inc	<p>Set the bits to control the source address increment. The address increment depends on the source data width as follows:</p> <p>Destination address increment. The address increment depends on the source data width as follows:</p> <p>Source data width = byte</p> <ul style="list-style-type: none"> <li>00b = byte</li> <li>01b = halfword</li> <li>10b = word</li> <li>11b = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains.</li> </ul> <p>Source data width = halfword</p> <ul style="list-style-type: none"> <li>00b = reserved</li> <li>01b = halfword</li> <li>10b = word</li> <li>11b = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains.</li> </ul> <p>Source data width = word</p> <ul style="list-style-type: none"> <li>00b = reserved</li> <li>01b = reserved</li> <li>10b = word</li> <li>11b = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains.</li> </ul>
[25:24]	src_size	<p>Set the bits to match the size of the source data:</p> <ul style="list-style-type: none"> <li>00b = byte</li> <li>01b = halfword</li> <li>10b = word</li> <li>11b = reserved</li> </ul>

**Table 9-13. channel\_cfg Bit Assignments (continued)**

Bit	Name	Description
[23:21]	dst_prot_ctrl	<p>Set the bits to control the state of HPROT[3:1] when the controller writes the destination data.</p> <p>Bit [23] Controls the state of HPROT[3] as follows:</p> <ul style="list-style-type: none"> <li>0 = HPROT[3] is LOW and the access is non-cacheable.</li> <li>1 = HPROT[3] is HIGH and the access is cacheable.</li> </ul> <p>Bit [22] Controls the state of HPROT[2] as follows:</p> <ul style="list-style-type: none"> <li>0 = HPROT[2] is LOW and the access is non-bufferable.</li> <li>1 = HPROT[2] is HIGH and the access is bufferable.</li> </ul> <p>Bit [21] Controls the state of HPROT[1] as follows:</p> <ul style="list-style-type: none"> <li>0 = HPROT[1] is LOW and the access is non-privileged.</li> <li>1 = HPROT[1] is HIGH and the access is privileged.</li> </ul>
[20:18]	src_prot_ctrl	<p>Set the bits to control the state of HPROT[3:1] when the controller reads the source data.</p> <p>Bit [20] Controls the state of HPROT[3] as follows:</p> <ul style="list-style-type: none"> <li>0 = HPROT[3] is LOW and the access is non-cacheable.</li> <li>1 = HPROT[3] is HIGH and the access is cacheable.</li> </ul> <p>Bit [19] Controls the state of HPROT[2] as follows:</p> <ul style="list-style-type: none"> <li>0 = HPROT[2] is LOW and the access is non-bufferable.</li> <li>1 = HPROT[2] is HIGH and the access is bufferable.</li> </ul> <p>Bit [18] Controls the state of HPROT[1] as follows:</p> <ul style="list-style-type: none"> <li>0 = HPROT[1] is LOW and the access is non-privileged.</li> <li>1 = HPROT[1] is HIGH and the access is privileged.</li> </ul>
[17:14]	R_power	<p>Set these bits to control how many DMA transfers can occur before the controller rearbitrates.</p> <p>The possible arbitration rate settings are:</p> <ul style="list-style-type: none"> <li>0000b = Arbitrates after each DMA transfer.</li> <li>0001b = Arbitrates after 2 DMA transfers.</li> <li>0010b = Arbitrates after 4 DMA transfers.</li> <li>0011b = Arbitrates after 8 DMA transfers.</li> <li>0100b = Arbitrates after 16 DMA transfers.</li> <li>0101b = Arbitrates after 32 DMA transfers.</li> <li>0110b = Arbitrates after 64 DMA transfers.</li> <li>0111b = Arbitrates after 128 DMA transfers.</li> <li>1000b = Arbitrates after 256 DMA transfers.</li> <li>1001b = Arbitrates after 512 DMA transfers.</li> <li>1010b-1111b = Arbitrates after 1024 DMA transfers. This means that no arbitration occurs during the DMA transfer because the maximum transfer size is 1024.</li> </ul>
[13:4]	n_minus_1	<p>Prior to the DMA cycle commencing, these bits represent the total number of DMA transfers that the DMA cycle contains. Set these bits according to the size of DMA cycle that is required.</p> <p>The 10-bit value indicates the number of DMA transfers, minus one. The possible values are:</p> <ul style="list-style-type: none"> <li>000000000b = 1 DMA transfer</li> <li>000000001b = 2 DMA transfers</li> <li>000000010b = 3 DMA transfer</li> <li>000000011b = 4 DMA transfers</li> <li>000000100b = 5 DMA transfers</li> <li>...</li> <li>111111111b = 1024 DMA transfers</li> </ul> <p>The controller updates this field immediately before entering the arbitration process. This enables the controller to store the number of outstanding DMA transfers that are necessary to complete the DMA cycle.</p>

**Table 9-13. channel\_cfg Bit Assignments (continued)**

Bit	Name	Description
[3]	next_useburst	<p>Controls if the chnl_useburst_set [C] bit is set to a 1, when the controller is performing a peripheral scatter-gather and is completing a DMA cycle that uses the alternate data structure.</p> <p>NOTE: Immediately before completion of the DMA cycle that the alternate data structure specifies, the controller sets the chnl_useburst_set [C] bit to 0 if the number of remaining transfers is less than <math>2^R</math>. The setting of the next_useburst bit controls if the controller performs an additional modification of the chnl_useburst_set [C] bit.</p> <p>In peripheral scatter-gather DMA cycle then after the DMA cycle that uses the alternate data structure completes, either:</p> <ul style="list-style-type: none"> <li>0 = the controller does not change the value of the chnl_useburst_set [C] bit. If the chnl_useburst_set [C] bit is 0 then for all the remaining DMA cycles in the peripheral scatter-gather transaction, the controller responds to requests on dma_req[] and dma_sreq[], when it performs a DMA cycle that uses an alternate data structure.</li> <li>1 = the controller sets the chnl_useburst_set [C] bit to a 1. Therefore, for the remaining DMA cycles in the peripheral scatter-gather transaction, the controller only responds to requests on dma_req[], when it performs a DMA cycle that uses an alternate data structure.</li> </ul>
[2:0]	cycle_ctrl	<p>The operating mode of the DMA cycle. The modes are:</p> <ul style="list-style-type: none"> <li>000b = Stop. Indicates that the data structure is invalid.</li> <li>001b = Basic. The controller must receive a new request before entering the arbitration process to enable the DMA cycle to complete.</li> <li>010b = Auto-request. The controller automatically inserts a request for the appropriate channel during the arbitration process. This means that the initial request is sufficient to enable the DMA cycle to complete.</li> <li>011b = Ping-pong. The controller performs a DMA cycle using one of the data structures. After the DMA cycle completes, it performs a DMA cycle using the other data structure. After the DMA cycle completes and provided that the host processor has updated the original data structure, it performs a DMA cycle using the original data structure. The controller continues to perform DMA cycles until it either reads an invalid data structure or the host processor changes the cycle_ctrl bits to 001b or 010b. See Ping-pong on page 2-23.</li> <li>100b = Memory scatter-gather (see <a href="#">Section 9.2.3.4.5</a>). When the controller operates in memory scatter-gather mode, use this value only in the primary data structure.</li> <li>101b = Memory scatter-gather (see <a href="#">Section 9.2.3.4.5</a>). When the controller operates in memory scatter-gather mode, use this value only in the alternate data structure.</li> <li>110b = Peripheral scatter-gather (see <a href="#">Section 9.2.3.4.6</a>). When the controller operates in peripheral scatter-gather mode, use this value only in the primary data structure.</li> <li>111b = Peripheral scatter-gather (see <a href="#">Section 9.2.3.4.6</a>). When the controller operates in peripheral scatter-gather mode, use this value only in the alternate data structure.</li> </ul>

At the start of a DMA cycle, or  $2^R$  DMA transfer, the controller fetches the channel\_cfg from system memory. After it performs  $2^R$ , or N, transfers, it stores the updated channel\_cfg in system memory.

The controller does not support a dst\_size value that is different from the src\_size value. If these values are not equal, the controller uses the src\_size value for source and destination and, when it next updates the n\_minus\_1 field, it also sets the dst\_size field to the same as the src\_size field.

After the controller completes the N transfers, it sets the cycle\_ctrl field to 000b to indicate that the channel\_cfg data is invalid. This prevents the controller from repeating the same DMA transfer.

#### 9.2.4.4 Address Calculation

To calculate the source address of a DMA transfer, the controller performs a left shift operation on the n\_minus\_1 value by a shift amount that src\_inc specifies, and then subtracts the resulting value from the source data end pointer. Similarly, to calculate the destination address of a DMA transfer, it performs a left shift operation on the n\_minus\_1 value by a shift amount that dst\_inc specifies, and then subtracts the resulting value from the destination end pointer.

Depending on the value of src\_inc and dst\_inc, the source address and destination address can be calculated using the equations:

$$\text{src\_inc} = 00b \text{ and } \text{dst\_inc} = 00b$$

- source address = src\_data\_end\_ptr - n\_minus\_1
- destination address = dst\_data\_end\_ptr - n\_minus\_1

`src_inc = 01b and dst_inc = 01b`

- source address = `src_data_end_ptr - (n_minus_1 << 1)`
- destination address = `dst_data_end_ptr - (n_minus_1 << 1)`

`src_inc = 10b and dst_inc = 10b`

- source address = `src_data_end_ptr - (n_minus_1 << 2)`
- destination address = `dst_data_end_ptr - (n_minus_1 << 2)`

`src_inc = 11b and dst_inc = 11b`

- source address = `src_data_end_ptr`
- destination address = `dst_data_end_ptr`

### 9.2.5 Peripheral Triggers

Up to eight trigger sources are multiplexed on each of the channels. The trigger sources are selected using the DMA\_CHn\_SRCCFG registers. For details about which trigger is mapped onto which channel, refer to the device-specific data sheet.

If any channel for the DMA controller is enabled and the channel receives a trigger from any of the peripherals, the trigger source is cleared when the DMA controller starts processing the channel.

In addition to the peripheral triggers, each channel can also be triggered by software by writing to the associated bit in the DMA Software Channel Trigger register.

---

**NOTE:** The DMA transfers are initiated upon rising edge of the peripheral triggers.

---

### 9.2.6 Interrupts

#### 9.2.6.1 DMA Completion Interrupts

The DMA controller outputs both the raw and a masked version of the channel's completion signal. The masked version feeds into the selection for the DMA\_INT1, DMA\_INT2, or DMA\_INT3, while the raw version feeds into the OR gate for DMA\_INT0.

If a channel is selected for DMA\_INT1, DMA\_INT2, or DMA\_INT3, it is masked from generating DMA\_INT0. DMA\_INT1, DMA\_INT2, and DMA\_INT3 are used for channels that require faster interrupt servicing, because the software overhead of finding out which channel completed is eliminated. In case of DMA\_INT0, the software must determine the channel that completed using the DMA\_INT0\_SRCFLG register.

---

**NOTE:** If the application maps the same source to two channels, multiple Active ack pulses are sent back to the source when these channels activate. Similarly, multiple interrupt events are generated. It is the responsibility of the application to not map a source to more than one enabled DMA channel.

---

#### 9.2.6.2 DMA Error Interrupt

In addition to the four interrupt lines, there is a dedicated interrupt line (DMA\_ERR) from the DMA controller to the CPU that is triggered when the DMA receives a bus error response during any transfer.

### 9.3 DMA Registers

DMA module is allocated a total of 8KB of addressable space for its registers. All register offset addresses not listed in these tables should be considered as reserved locations and the register contents should not be modified.

**Table 9-14. DMA Registers**

Offset	Acronym	Register Name	Type	Reset	Section
000h	DMA_DEVICE_CFG	Device Configuration Status Register	R	0000nnnn <sup>(1)</sup>	<a href="#">Section 9.3.1</a>
004h	DMA_SW_CHTRIG	Software Channel Trigger Register	RW	0h	<a href="#">Section 9.3.2</a>
010h + n*4h	DMA_CHn_SRCCFG (n = 0 to NUM_DMA_CHANNELS)	Channel n Source Configuration Register	RW	0h	<a href="#">Section 9.3.3</a>
100h	DMA_INT1_SRCCFG	Interrupt 1 Source Channel Configuration Register	RW	0h	<a href="#">Section 9.3.4</a>
104h	DMA_INT2_SRCCFG	Interrupt 2 Source Channel Configuration Register	RW	0h	<a href="#">Section 9.3.5</a>
108h	DMA_INT3_SRCCFG	Interrupt 3 Source Channel Configuration Register	RW	0h	<a href="#">Section 9.3.6</a>
110h	DMA_INT0_SRCFLG	Interrupt 0 Source Channel Flag Register	RW	0h	<a href="#">Section 9.3.7</a>
114h	DMA_INT0_CLRFLG	Interrupt 0 Source Channel Clear Flag Register	W	-	<a href="#">Section 9.3.8</a>
1000h	DMA_STAT	Status Register	R	0x-0nn0000 <sup>(2)</sup>	<a href="#">Section 9.3.9</a>
1004h	DMA_CFG	Configuration Register	W	-	<a href="#">Section 9.3.10</a>
1008h	DMA_CTLBASE	Channel Control Data Base Pointer Register	RW	0h	<a href="#">Section 9.3.11</a>
100Ch	DMA_ALTBASE	Channel Alternate Control Data Base Pointer Register	R	000000nnh <sup>(3)</sup>	<a href="#">Section 9.3.12</a>
1010h	DMA_WAITSTAT	Channel Wait on Request Status Register	R	0h	<a href="#">Section 9.3.13</a>
1014h	DMA_SWREQ	Channel Software Request Register	W	-	<a href="#">Section 9.3.14</a>
1018h	DMA_USEBURSTSET	Channel Useburst Set Register	RW	0h	<a href="#">Section 9.3.15</a>
101Ch	DMA_USEBURSTCLR	Channel Useburst Clear Register	W	-	<a href="#">Section 9.3.16</a>
1020h	DMA_REQMASKSET	Channel Request Mask Set Register	RW	0h	<a href="#">Section 9.3.17</a>
1024h	DMA_REQMASKCLR	Channel Request Mask Clear Register	W	-	<a href="#">Section 9.3.18</a>
1028h	DMA_ENASET	Channel Enable Set Register	RW	0h	<a href="#">Section 9.3.19</a>
102Ch	DMA_ENACLR	Channel Enable Clear Register	W	-	<a href="#">Section 9.3.20</a>
1030h	DMA_ALTSET	Channel Primary-Alternate Set Register	RW	0h	<a href="#">Section 9.3.21</a>
1034h	DMA_ALTCLR	Channel Primary-Alternate Clear Register	W	-	<a href="#">Section 9.3.22</a>
1038h	DMA_PRIOSET	Channel Priority Set Register	RW	0h	<a href="#">Section 9.3.23</a>
103Ch	DMA_PRIOCLR	Channel Priority Clear Register	W	-	<a href="#">Section 9.3.24</a>
104Ch	DMA_ERRCLR	Bus Error Clear Register	RW	0h	<a href="#">Section 9.3.25</a>

<sup>(1)</sup> The reset value depends on the number of DMA channels available in the controller and number of trigger sources per channel.

<sup>(2)</sup> The reset value depends on the number of DMA channels available and if the integration test logic is included.

<sup>(3)</sup> The reset value depends on the number of DMA channels available.

### 9.3.1 DMA\_DEVICE\_CFG Register (offset = 000h)

DMA Device Configuration status Register

**Figure 9-11. DMA\_DEVICE\_CFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUM_SRC_PER_CHANNEL								NUM_DMA_CHANNELS							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

**Table 9-15. DMA\_DEVICE\_CFG Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Reads return 0h
15-8	NUM_SRC_PER_CH ANNEL <sup>(1)</sup>	R	Undefined	Reflects the number of DMA sources per channel
7-0	NUM_DMA_CHANNELS <sup>(1)</sup>	R	Undefined	Reflects the number of DMA channels available on the device

<sup>(1)</sup> Refer to appropriate device datasheet for the value in this field

### 9.3.2 DMA\_SW\_CHTRIG Register (offset = 004h)

DMA Software Channel Trigger Register

**Figure 9-12. DMA\_SW\_CHTRIG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
rw-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
rw-0															

**Table 9-16. DMA\_SW\_CHTRIG Register Description**

Bit	Field	Type	Reset	Description
31	CH31	RW	0h	Write 1 triggers DMA_CHANNEL31. Bit is auto-cleared when channel goes active.
30	CH30	RW	0h	Write 1 triggers DMA_CHANNEL30. Bit is auto-cleared when channel goes active.
29	CH29	RW	0h	Write 1 triggers DMA_CHANNEL29. Bit is auto-cleared when channel goes active.
28	CH28	RW	0h	Write 1 triggers DMA_CHANNEL28. Bit is auto-cleared when channel goes active.
27	CH27	RW	0h	Write 1 triggers DMA_CHANNEL27. Bit is auto-cleared when channel goes active.
26	CH26	RW	0h	Write 1 triggers DMA_CHANNEL26. Bit is auto-cleared when channel goes active.
25	CH25	RW	0h	Write 1 triggers DMA_CHANNEL25. Bit is auto-cleared when channel goes active.
24	CH24	RW	0h	Write 1 triggers DMA_CHANNEL24. Bit is auto-cleared when channel goes active.
23	CH23	RW	0h	Write 1 triggers DMA_CHANNEL23. Bit is auto-cleared when channel goes active.
22	CH22	RW	0h	Write 1 triggers DMA_CHANNEL22. Bit is auto-cleared when channel goes active.
21	CH21	RW	0h	Write 1 triggers DMA_CHANNEL21. Bit is auto-cleared when channel goes active.
20	CH20	RW	0h	Write 1 triggers DMA_CHANNEL20. Bit is auto-cleared when channel goes active.
19	CH19	RW	0h	Write 1 triggers DMA_CHANNEL19. Bit is auto-cleared when channel goes active.
18	CH18	RW	0h	Write 1 triggers DMA_CHANNEL18. Bit is auto-cleared when channel goes active.
17	CH17	RW	0h	Write 1 triggers DMA_CHANNEL17. Bit is auto-cleared when channel goes active.
16	CH16	RW	0h	Write 1 triggers DMA_CHANNEL16. Bit is auto-cleared when channel goes active.
15	CH15	RW	0h	Write 1 triggers DMA_CHANNEL15. Bit is auto-cleared when channel goes active.
14	CH14	RW	0h	Write 1 triggers DMA_CHANNEL14. Bit is auto-cleared when channel goes active.
13	CH13	RW	0h	Write 1 triggers DMA_CHANNEL13. Bit is auto-cleared when channel goes active.
12	CH12	RW	0h	Write 1 triggers DMA_CHANNEL12. Bit is auto-cleared when channel goes active.
11	CH11	RW	0h	Write 1 triggers DMA_CHANNEL11. Bit is auto-cleared when channel goes active.
10	CH10	RW	0h	Write 1 triggers DMA_CHANNEL10. Bit is auto-cleared when channel goes active.

**Table 9-16. DMA\_SW\_CHTRIG Register Description (continued)**

Bit	Field	Type	Reset	Description
9	CH9	RW	0h	Write 1 triggers DMA_CHANNEL9. Bit is auto-cleared when channel goes active.
8	CH8	RW	0h	Write 1 triggers DMA_CHANNEL8. Bit is auto-cleared when channel goes active.
7	CH7	RW	0h	Write 1 triggers DMA_CHANNEL7. Bit is auto-cleared when channel goes active.
6	CH6	RW	0h	Write 1 triggers DMA_CHANNEL6. Bit is auto-cleared when channel goes active.
5	CH5	RW	0h	Write 1 triggers DMA_CHANNEL5. Bit is auto-cleared when channel goes active.
4	CH4	RW	0h	Write 1 triggers DMA_CHANNEL4. Bit is auto-cleared when channel goes active.
3	CH3	RW	0h	Write 1 triggers DMA_CHANNEL3. Bit is auto-cleared when channel goes active.
2	CH2	RW	0h	Write 1 triggers DMA_CHANNEL2. Bit is auto-cleared when channel goes active.
1	CH1	RW	0h	Write 1 triggers DMA_CHANNEL1. Bit is auto-cleared when channel goes active.
0	CH0	RW	0h	Write 1 triggers DMA_CHANNEL0. Bit is auto-cleared when channel goes active.

**NOTE:** If the number of channels is less than 32, all bits for channels that are not implemented will behave as reserved.

**NOTE:** If a channel x is triggered using software, the DMA controller will override/mask the DMA active/acknowledge pulse that would ideally be sent to the source mapped to that channel (set through the register DMA\_CHx\_SRCCFG). It is the application's responsibility to handle the completion event in such scenarios, because if a DMA request flag is also set in the source, it will need to be cleared explicitly by software.

The intent of providing a software trigger feature is to either allow software to completely control the DMA channel, or to emulate the DMA request/acknowledge functionality of the actual source that is mapped to that channel.

### **9.3.3 DMA\_CHn\_SRCCFG Register (offset = 010h + 4h\*n, n = 0 through NUM\_DMA\_CHANNELS)**

DMA Channel n Source Configuration Register (n = 0 through Number of DMA channels)

**Figure 9-13. DMA\_CHn\_SRCCFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	rw-0						

**Table 9-17. DMA\_CHn\_SRCCFG Register Description**

Bit	Field	Type	Reset	Description
31-8	Reserved	R	0h	Reserved. Reads return 0h
7-0	DMA_SRC	RW	0h	Controls which device level DMA source is mapped to the channel input (bits higher than the number of available sources will be forced to r mode)

### 9.3.4 DMA\_INT1\_SRCCFG Register (offset = 100h)

DMA Interrupt 1 Source Channel Configuration Register

**Figure 9-14. DMA\_INT1\_SRCCFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EN	INT_SRC				
r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 9-18. DMA\_INT1\_SRCCFG Register Description**

Bit	Field	Type	Reset	Description
31-6	Reserved	R	0h	Reserved. Reads return 0h
5	EN <sup>(1)</sup>	RW	0h	When 1, enables the DMA_INT1 mapping.
4-0	INT_SRC <sup>(1)(2)</sup>	RW	0h	Controls which channel's completion event is mapped as a source of this Interrupt (bits higher than number of channels will be forced to r)

<sup>(1)</sup> Enabling DMA\_INT1 mapping and selecting a particular channel completion to map to this interrupt will result in the completion being masked from generating INT0

<sup>(2)</sup> If the number of channels is less than 32, all bits for channels that are not implemented should behave as reserved.

### 9.3.5 DMA\_INT2\_SRCCFG Register (offset = 104h)

DMA Interrupt 2 Source Channel Configuration Register

**Figure 9-15. DMA\_INT2\_SRCCFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EN	INT_SRC				
r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 9-19. DMA\_INT2\_SRCCFG Register Description**

Bit	Field	Type	Reset	Description
31-6	Reserved	R	0h	Reserved. Reads return 0h
5	EN <sup>(1)</sup>	RW	0h	When 1, enables the DMA_INT2 mapping.
4-0	INT_SRC <sup>(1)(2)</sup>	RW	0h	Controls which channel's completion event is mapped as a source of this Interrupt (bits higher than number of channels will be forced to r)

<sup>(1)</sup> Enabling DMA\_INT2 mapping and selecting a particular channel completion to map to this interrupt will result in the completion being masked from generating INT0

<sup>(2)</sup> If the number of channels is less than 32, all bits for channels that are not implemented should behave as reserved.

### 9.3.6 DMA\_INT3\_SRCCFG Register (offset = 108h)

DMA Interrupt 3 Source Channel Configuration Register

**Figure 9-16. DMA\_INT3\_SRCCFG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EN	INT_SRC				
r	r	r	r	r	r	r	r	r	r	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 9-20. DMA\_INT3\_SRCCFG Register Description**

Bit	Field	Type	Reset	Description
31-6	Reserved	R	0h	Reserved. Always returns 0h
5	EN <sup>(1)</sup>	RW	0h	When 1, enables the DMA_INT3 mapping.
4-0	INT_SRC <sup>(2)(3)</sup>	RW	0h	Controls which channel's completion event is mapped as a source of this interrupt

<sup>(1)</sup> Enabling DMA\_INT1 mapping and selecting a particular channel completion to map to this interrupt will result in the completion being masked from generating INT0

<sup>(2)</sup> Enabling DMA\_INT3 mapping and selecting a particular channel completion to map to this interrupt will result in the completion being masked from generating INT0

<sup>(3)</sup> If the number of channels is less than 32, all bits for channels that are not implemented should behave as reserved.

### 9.3.7 DMA\_INT0\_SRCFLG Register (offset = 110h)

DMA Interrupt 0 Source Channel Flag Register

**Figure 9-17. DMA\_INT0\_SRCFLG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
r-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
r-0															

**Table 9-21. DMA\_INT0\_SRCFLG Register Description**

Bit	Field	Type	Reset	Description
31	CH31	R	0h	If 1, indicates that Channel 31 was the source of DMA_INT0
30	CH30	R	0h	If 1, indicates that Channel 30 was the source of DMA_INT0
29	CH29	R	0h	If 1, indicates that Channel 29 was the source of DMA_INT0
28	CH28	R	0h	If 1, indicates that Channel 28 was the source of DMA_INT0
27	CH27	R	0h	If 1, indicates that Channel 27 was the source of DMA_INT0
26	CH26	R	0h	If 1, indicates that Channel 26 was the source of DMA_INT0
25	CH25	R	0h	If 1, indicates that Channel 25 was the source of DMA_INT0
24	CH24	R	0h	If 1, indicates that Channel 24 was the source of DMA_INT0
23	CH23	R	0h	If 1, indicates that Channel 23 was the source of DMA_INT0
22	CH22	R	0h	If 1, indicates that Channel 22 was the source of DMA_INT0
21	CH21	R	0h	If 1, indicates that Channel 21 was the source of DMA_INT0
20	CH20	R	0h	If 1, indicates that Channel 20 was the source of DMA_INT0
19	CH19	R	0h	If 1, indicates that Channel 19 was the source of DMA_INT0
18	CH18	R	0h	If 1, indicates that Channel 18 was the source of DMA_INT0
17	CH17	R	0h	If 1, indicates that Channel 17 was the source of DMA_INT0
16	CH16	R	0h	If 1, indicates that Channel 16 was the source of DMA_INT0
15	CH15	R	0h	If 1, indicates that Channel 15 was the source of DMA_INT0
14	CH14	R	0h	If 1, indicates that Channel 14 was the source of DMA_INT0
13	CH13	R	0h	If 1, indicates that Channel 13 was the source of DMA_INT0
12	CH12	R	0h	If 1, indicates that Channel 12 was the source of DMA_INT0
11	CH11	R	0h	If 1, indicates that Channel 11 was the source of DMA_INT0
10	CH10	R	0h	If 1, indicates that Channel 10 was the source of DMA_INT0
9	CH9	R	0h	If 1, indicates that Channel 9 was the source of DMA_INT0
8	CH8	R	0h	If 1, indicates that Channel 8 was the source of DMA_INT0
7	CH7	R	0h	If 1, indicates that Channel 7 was the source of DMA_INT0
6	CH6	R	0h	If 1, indicates that Channel 6 was the source of DMA_INT0
5	CH5	R	0h	If 1, indicates that Channel 5 was the source of DMA_INT0
4	CH4	R	0h	If 1, indicates that Channel 4 was the source of DMA_INT0
3	CH3	R	0h	If 1, indicates that Channel 3 was the source of DMA_INT0
2	CH2	R	0h	If 1, indicates that Channel 2 was the source of DMA_INT0
1	CH1	R	0h	If 1, indicates that Channel 1 was the source of DMA_INT0
0	CH0	R	0h	If 1, indicates that Channel 0 was the source of DMA_INT0

**NOTE:** If the number of channels is less than 32, all bits for channels that are not implemented should behave as reserved.

### 9.3.8 DMA\_INT0\_CLRFLG Register (offset = 114h)

DMA Interrupt 0 Source Channel Clear Flag Register

**Figure 9-18. DMA\_INT0\_CLRFLG Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

**Table 9-22. DMA\_INT0\_CLRFLG Register Description**

Bit	Field	Type	Reset	Description
31	CH31	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
30	CH30	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
29	CH29	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
28	CH28	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
27	CH27	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
26	CH26	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
25	CH25	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
24	CH24	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
23	CH23	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
22	CH22	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
21	CH21	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
20	CH20	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
19	CH19	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
18	CH18	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
17	CH17	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
16	CH16	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
15	CH15	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
14	CH14	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
13	CH13	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
12	CH12	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
11	CH11	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
10	CH10	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect

**Table 9-22. DMA\_INT0\_CLRFLG Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
9	CH9	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
8	CH8	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
7	CH7	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
6	CH6	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
5	CH5	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
4	CH4	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
3	CH3	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
2	CH2	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
1	CH1	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect
0	CH0	W	NA	Write 1 clears the corresponding flag bit in the DMA_INT0_SRCFLG register. Write 0 has no effect

**NOTE:** If the number of channels is less than 32, all bits for channels that are not implemented should behave as reserved.

### 9.3.9 DMA\_STAT Register (offset = 1000h) [reset = 0h]

DMA Status Register. The read-only DMA\_STAT register returns the status of the controller. You cannot read this register when the controller is in the reset state.

**Figure 9-19. DMA\_STAT Register**

31	30	29	28	27	26	25	24
TESTSTAT				RESERVED			
r				r-0			
23	22	21	20	19	18	17	16
RESERVED			DMACHANS				
r-0				r			
15	14	13	12	11	10	9	8
RESERVED							
r-0				r-0			
7	6	5	4	3	2	1	0
STATE				RESERVED			MASTEN
r				r-0			r

**Table 9-23. DMA\_STAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-28	TESTSTAT	R	X	To reduce the gate count you can configure the controller to exclude the integration test logic. 2h-Fh = Reserved 0h = Controller does not include the integration test logic 1h = Controller includes the integration test logic
27-21	RESERVED	R	0h	Reserved
20-16	DMACHANS	R	X	Number of available DMA channels minus one. 0000b = Controller configured to use 1 DMA channel 0001b = Controller configured to use 2 DMA channels 1110b = Controller configured to use 31 DMA channels 1111b = Controller configured to use 32 DMA channels
15-8	RESERVED	R	0h	Reserved
7-4	STATE	R	X	Current state of the control state machine. State can be one of the following: 0b = idle 1b = reading channel controller data 10b = reading source data end pointer 11b = reading destination data end pointer 100b = reading source data 101b = writing destination data 110b = waiting for DMA request to clear 111b = writing channel controller data 1000b = stalled 1001b = done 1010b = peripheral scatter-gather transition 1011b = Reserved 1100b = Reserved 1101b = Reserved 1110b = Reserved 1111b = Reserved
3-1	RESERVED	R	0h	Reserved
0	MASTEN	R	X	Enable status of the controller 0b = Controller disabled 1b = Controller enabled

### 9.3.10 DMA\_CFG Register (offset = 1004h) [reset = 0h]

DMA Configuration Register. The write-only DMA\_CFG Register controls the configuration of the controller.

**Figure 9-20. DMA\_CFG Register**

31	30	29	28	27	26	25	24
RESERVED							
r-0							
23	22	21	20	19	18	17	16
RESERVED							
r-0							
15	14	13	12	11	10	9	8
RESERVED							
r-0							
7	6	5	4	3	2	1	0
CHPROTCTRL			RESERVED				MASTEN
w-0			r-0				w-0

**Table 9-24. DMA\_CFG Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reserved
7-5	CHPROTCTRL	W	0h	Sets the AHB-Lite protection by controlling the HPROT[3:1] signal levels as follows: Bit [7] Controls HPROT[3] to indicate if a cacheable access is occurring. Bit [6] Controls HPROT[2] to indicate if a bufferable access is occurring. Bit [5] Controls HPROT[1] to indicate if a privileged access is occurring. Note: When bit [n] = 1 then the corresponding HPROT is HIGH. When bit [n] = 0 then the corresponding HPROT is LOW.
4-1	RESERVED	R	0h	Reserved
0	MASTEN	W	0h	Enable status of the controller 0b = Controller disabled 1b = Controller enabled

### 9.3.11 DMA\_CTLBASE Register (offset = 1008h) [reset = 0h]

DMA Channel Control Data Base Pointer Register. The DMA\_CTLBASE Register is a read/write register. You must configure this register so that the base pointer points to a location in your system memory.

Note: The controller provides no internal memory for storing the channel control data structure. The amount of system memory that you must assign to the controller depends on the number of DMA channels and whether you configure it to use the alternate data structure. Therefore, the base pointer address requires a variable number of bits that depend on the system implementation. You cannot read this register when the controller is in the reset state.

**Figure 9-21. DMA\_CTLBASE Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16						
ADDR																					
rw																					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
ADDR										RESERVED											
rw																					
r-0																					

**Table 9-25. DMA\_CTLBASE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-5	ADDR	RW	X	Pointer to the base address of the primary data structure.
4-0	RESERVED	R	0h	Reserved

### **9.3.12 DMA\_ALTBASE Register (offset = 100Ch) [reset = 0h]**

DMA Channel Alternate Control Data Base Pointer Register. The Channel alternate control data base pointer register returns the base address of the alternate data structure. You cannot read this register when the controller is in the reset state.

This register removes the necessity for application software to calculate the base address of the alternate data structure.

**Figure 9-22. DMA\_ALTBASE Register**

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
ADDR																															
r																															

**Table 9-26. DMA\_ALTBASE Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	ADDR	R	X	Base address of the alternate data structure

### 9.3.13 DMA\_WAITSTAT Register (offset = 1010h) [reset = 0h]

DMA Channel Wait on Request Status Register. The Channel wait on request status register returns the status of `dma_waitonreq[]`. You cannot read this register when the controller is in the reset state.

**Figure 9-23. DMA\_WAITSTAT Register**

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
WAITREQ																															
r=0																															

**Table 9-27. DMA\_WAITSTAT Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	WAITREQ	R	0h	Channel wait on request status. Read as: Bit [C] = 0 <code>dma_waitonreq[C]</code> is LOW. Bit [C] = 1 <code>dma_waitonreq[C]</code> is HIGH.

### **9.3.14 DMA\_SWREQ Register (offset = 1014h) [reset = 0h]**

DMA Channel Software Request Register. The Channel software request register enables you to generate a software DMA request.

**Figure 9-24. DMA\_SWREQ Register**

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
CHNL_SW_REQ																															
w-0																															

**Table 9-28. DMA\_SWREQ Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CHNL_SW_REQ	W	0h	<p>Set the appropriate bit to generate a software DMA request on the corresponding DMA channel.</p> <p>Write as: Bit [C] = 0 Does not create a DMA request for channel C.</p> <p>Bit [C] = 1 Creates a DMA request for channel C.</p> <p>Writing to a bit where a DMA channel is not implemented does not create a DMA request for that channel.</p>

### 9.3.15 DMA\_USEBURSTSET Register (offset = 1018h) [reset = 0h]

DMA Channel Useburst Set Register. The Channel useburst set register disables the single request `dma_sreq[]` input from generating requests, and therefore only the request, `dma_req[]`, generates requests. Reading the register returns the useburst status.

**Figure 9-25. DMA\_USEBURSTSET Register**

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
SET																															
rw-0																															

**Table 9-29. DMA\_USEBURSTSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SET	RW	0h	<p>Returns the useburst status, or disables <code>dma_sreq[C]</code> from generating DMA requests.            Read as: Bit [C] = 0 DMA channel C responds to requests that it receives on <code>dma_req[C]</code> or <code>dma_sreq[C]</code>.            The controller performs <math>2^R</math>, or single, bus transfers.            Bit [C] = 1 DMA channel C does not respond to requests that it receives on <code>dma_sreq[C]</code>.            The controller only responds to <code>dma_req[C]</code> requests and performs <math>2^R</math> transfers.            Write as: Bit [C] = 0 No effect.            Use the DMA_USEBURST_CLR Register to set bit [C] to 0.            Bit [C] = 1 Disables <code>dma_sreq[C]</code> from generating DMA requests.            The controller performs <math>2^R</math> transfers.            Writing to a bit where a DMA channel is not implemented has no effect.</p>

### **9.3.16 DMA\_USEBURSTCLR Register (offset = 101Ch) [reset = 0h]**

DMA Channel Useburst Clear Register. The Channel useburst clear register enables dma\_sreq[] to generate requests.

**Figure 9-26. DMA\_USEBURSTCLR Register**

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
CLR																															
w-0																															

**Table 9-30. DMA\_USEBURSTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLR	W	0h	<p>Set the appropriate bit to enable dma_sreq[] to generate requests.            Write as: Bit [C] = 0 No effect.            Use the DMA_USEBURST_SET Register to disable dma_sreq[] from generating requests.            Bit [C] = 1 Enables dma_sreq[C] to generate DMA requests.            Writing to a bit where a DMA channel is not implemented has no effect.</p>

### 9.3.17 DMA\_REQMASKSET Register (offset = 1020h) [reset = 0h]

DMA Channel Request Mask Set Register. The Channel request mask set register disables a HIGH on `dma_req[]`, or `dma_sreq[]`, from generating a request. Reading the register returns the request mask status for `dma_req[]` and `dma_sreq[]`.

**Figure 9-27. DMA\_REQMASKSET Register**

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
SET																															
rw-0																															

**Table 9-31. DMA\_REQMASKSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SET	RW	0h	Returns the request mask status of <code>dma_req[]</code> and <code>dma_sreq[]</code> , or disables the corresponding channel from generating DMA requests. Read as: Bit [C] = 0 External requests are enabled for channel C. Bit [C] = 1 External requests are disabled for channel C. Write as: Bit [C] = 0 No effect. Use the DMA_REQMASKCLR Register to enable DMA requests. Bit [C] = 1 Disables <code>dma_req[C]</code> and <code>dma_sreq[C]</code> from generating DMA requests. Writing to a bit where a DMA channel is not implemented has no effect.

### **9.3.18 DMA\_REQMASKCLR Register (offset = 1024h) [reset = 0h]**

DMA Channel Request Mask Clear Register. The Channel request mask clear register enables a HIGH on dma\_req[], or dma\_sreq[], to generate a request.

**Figure 9-28. DMA\_REQMASKCLR Register**

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
CLR																															
w-0																															

**Table 9-32. DMA\_REQMASKCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLR	W	0h	<p>Set the appropriate bit to enable DMA requests for the channel corresponding to dma_req[] and dma_sreq[].</p> <p>Write as: Bit [C] = 0 No effect.</p> <p>Use the DMA_REQMASKSET Register to disable dma_req[] and dma_sreq[] from generating requests.</p> <p>Bit [C] = 1 Enables dma_req[C] or dma_sreq[C] to generate DMA requests.</p> <p>Writing to a bit where a DMA channel is not implemented has no effect.</p>

### 9.3.19 DMA\_ENASET Register (offset = 1028h) [reset = 0h]

DMA Channel Enable Set Register. The Channel enable set register enables you to enable a DMA channel. Reading the register returns the enable status of the channels.

**Figure 9-29. DMA\_ENASET Register**

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
SET																															
rw-0																															

**Table 9-33. DMA\_ENASET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SET	RW	0h	Returns the enable status of the channels, or enables the corresponding channels. Read as: Bit [C] = 0 Channel C is disabled. Bit [C] = 1 Channel C is enabled. Write as: Bit [C] = 0 No effect. Use the DMA_ENACLR Register to disable a channel. Bit [C] = 1 Enables channel C. Writing to a bit where a DMA channel is not implemented has no effect.

### **9.3.20 DMA\_ENACLR Register (offset = 102Ch) [reset = 0h]**

DMA Channel Enable Clear Register. The Channel enable clear register enables you to disable a DMA channel.

chnl\_enable\_clr is shown in [Figure 9-30](#) and described in [Table 9-34](#).

**Figure 9-30. DMA\_ENACLR Register**

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
chnl_enable_clr																															
w-0																															

**Table 9-34. DMA\_ENACLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLR	W	0h	<p>Set the appropriate bit to disable the corresponding DMA channel.            Write as: Bit [C] = 0 No effect.            Use the DMA_ENASET Register to enable DMA channels.            Bit [C] = 1 Disables channel C.            Writing to a bit where a DMA channel is not implemented has no effect.            Note: The controller disables a channel, by setting the appropriate bit, when:</p> <ul style="list-style-type: none"> <li>a) it completes the DMA cycle</li> <li>b) it reads a channel_cfg memory location which has cycle_ctrl = b000</li> <li>c) an ERROR occurs on the AHB-Lite bus.</li> </ul>

### 9.3.21 DMA\_ALTSET Register (offset = 1030h) [reset = 0h]

DMA Channel Primary-Alternate Set Register. The Channel primary-alternate set register enables you to configure a DMA channel to use the alternate data structure. Reading the register returns the status of which data structure is in use for the corresponding DMA channel.

**Figure 9-31. DMA\_ALTSET Register**

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
SET																															
rw-0																															

**Table 9-35. DMA\_ALTSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SET	RW	0h	Returns the channel control data structure status, or selects the alternate data structure for the corresponding DMA channel. Read as: Bit [C] = 0 DMA channel C is using the primary data structure. Bit [C] = 1 DMA channel C is using the alternate data structure. Write as: Bit [C] = 0 No effect. Use the DMA_ALTCLR Register to set bit [C] to 0. Bit [C] = 1 Selects the alternate data structure for channel C. Writing to a bit where a DMA channel is not implemented has no effect. Note: The controller toggles the value of the chnl_pri_alt_set [C] bit after it completes: the four transfers that the primary data structure specifies for a memory scatter-gather, or peripheral scatter-gather, DMA cycle all the transfers that the primary data structure specifies for a ping-pong DMA cycle all the transfers that the alternate data structure specifies for the following DMA cycle types: ping-pong, memory scatter-gather, or peripheral scatter-gather.

### 9.3.22 DMA\_ALTCLR Register (offset = 1034h) [reset = 0h]

DMA Channel Primary-Alternate Clear Register. The Channel primary-alternate clear register enables you to configure a DMA channel to use the primary data structure.

**Figure 9-32. DMA\_ALTCLR Register**

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
CLR																															
w-0																															

**Table 9-36. DMA\_ALTCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLR	W	0h	<p>Set the appropriate bit to select the primary data structure for the corresponding DMA channel.          Write as: Bit [C] = 0 No effect.          Use the DMA_ALTSET Register to select the alternate data structure.          Bit [C] = 1 Selects the primary data structure for channel C.          Writing to a bit where a DMA channel is not implemented has no effect.          Note: The controller toggles the value of the chnl_pri_alt_clr [C] bit after it completes:          the four transfers that the primary data structure specifies for a memory scatter-gather, or peripheral scatter-gather, DMA cycle          all the transfers that the primary data structure specifies for a ping-pong DMA cycle          all the transfers that the alternate data structure specifies for the following DMA cycle types: ping-pong, memory scatter-gather, or peripheral scatter-gather.</p>

### 9.3.23 DMA\_PRIOSSET Register (offset = 1038h) [reset = 0h]

DMA Channel Priority Set Register. The Channel priority set register enables you to configure a DMA channel to use the high priority level. Reading the register returns the status of the channel priority mask.

**Figure 9-33. DMA\_PRIOSSET Register**

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
SET																															
rw-0																															

**Table 9-37. DMA\_PRIOSSET Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	SET	RW	0h	Returns the channel priority mask status, or sets the channel priority to high. Read as: Bit [C] = 0 DMA channel C is using the default priority level. Bit [C] = 1 DMA channel C is using a high priority level. Write as: Bit [C] = 0 No effect. Use the DMA_PRIOCLR Register to set channel C to the default priority level. Bit [C] = 1 Channel C uses the high priority level. Writing to a bit where a DMA channel is not implemented has no effect.

### **9.3.24 DMA\_PRIOCLR Register (offset = 103Ch) [reset = 0h]**

DMA Channel Priority Clear Register. The Channel priority clear register enables you to configure a DMA channel to use the default priority level.

**Figure 9-34. DMA\_PRIOCLR Register**

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
CLR																															
w-0																															

**Table 9-38. DMA\_PRIOCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-0	CLR	W	0h	<p>Set the appropriate bit to select the default priority level for the specified DMA channel.</p> <p>Write as: Bit [C] = 0 No effect.</p> <p>Use the DMA_PRIOSSET Register to set channel C to the high priority level.</p> <p>Bit [C] = 1 Channel C uses the default priority level.</p> <p>Writing to a bit where a DMA channel is not implemented has no effect.</p>

### 9.3.25 DMA\_ERRCLR Register (offset = 104Ch) [reset = 0h]

DMA Bus Error Clear Register. The Bus error clear register returns the status of dma\_err, and enables you to set dma\_err LOW.

**Figure 9-35. DMA\_ERRCLR Register**

31	30	29	28	27	26	25	24
RESERVED							
r-0							
23	22	21	20	19	18	17	16
RESERVED							
r-0							
15	14	13	12	11	10	9	8
RESERVED							
r-0							
7	6	5	4	3	2	1	0
RESERVED							
r-0							
ERRCLR							
rw-0							

**Table 9-39. DMA\_ERRCLR Register Field Descriptions**

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reserved
0	ERRCLR	RW	0h	Returns the status of dma_err, or sets the signal LOW. Read as: 0 = dma_err is LOW 1 = dma_err is HIGH. Write as: 0 = No effect, status of dma_err is unchanged. 1 = Sets dma_err LOW. For test purposes, use the ERRSET register to set dma_err HIGH. Note: If you deassert dma_err at the same time as an ERROR occurs on the AHB-Lite bus, then the ERROR condition takes precedence and dma_err remains asserted.

**Digital I/O**

This chapter describes the operation of the digital I/O ports in all devices.

Topic	Page
10.1 Digital I/O Introduction .....	498
10.2 Digital I/O Operation .....	499
10.3 I/O Configuration .....	502
10.4 Digital I/O Registers .....	504

## 10.1 Digital I/O Introduction

The digital I/O features include:

- Independently programmable individual I/Os
- Any combination of input or output
- Individually configurable interrupts for ports (available for certain ports only)
- Independent input and output data registers
- Individually configurable pullup or pulldown resistors
- Wake-up capability from ultra-low-power modes (available for certain ports only)
- Individually configurable high drive I/Os (available for certain I/Os only)

Devices within the family may have up to eleven digital I/O ports implemented (P1 to P10 and PJ). Most ports contain eight I/O lines; however, some ports may contain less (see the device-specific data sheet for ports available). Each I/O line is individually configurable for input or output direction, and each can be individually read or written. Each I/O line is individually configurable for pullup or pulldown resistors.

Certain ports have interrupt and wake-up capability from ultra-low-power modes (see device-specific data sheet for ports with interrupt and wake-up capability). Each interrupt can be individually enabled and configured to provide an interrupt on a rising or falling edge of an input signal. All interrupts are fed into an encoded interrupt vector register, allowing the application to determine which pin of a port has generated the event.

Individual ports can be accessed as byte-wide ports or can be combined into half-word-wide ports. Port pairs P1 and P2, P3 and P4, P5 and P6, P7 and P8, and so on, are associated with the names PA, PB, PC, PD, and so on, respectively. All port registers are handled in this manner with this naming convention. The main exception are the interrupt vector registers, for example, interrupts for ports P1 and P2 must be handled through P1IV and P2IV, PAIV does not exist.

When writing to port PA with half-word operations, all 16 bits are written to the port. When writing to the lower byte of port PA using byte operations, the upper byte remains unchanged. Similarly, writing to the upper byte of port PA using byte instructions leaves the lower byte unchanged. When writing to a port that contains fewer than the maximum number of bits possible, the unused bits are don't care. Ports PB, PC, PD, PE, and PF behave similarly.

Reading port PA using half-word operations causes all 16 bits to be transferred to the destination. Reading the lower or upper byte of port PA (P1 or P2) and storing to memory using byte operations causes only the lower or upper byte to be transferred to the destination, respectively. When reading from ports that contain fewer than the maximum bits possible, unused bits are read as zeros (similarly for port PJ).

## 10.2 Digital I/O Operation

The digital I/O are configured with user software. The setup and operation of the digital I/O are discussed in the following sections.

### 10.2.1 Input Registers (*PxIN*)

Each bit in each PxIN register reflects the value of the input signal at the corresponding I/O pin when the pin is configured as I/O function. These registers are read only.

- Bit = 0: Input is low
- Bit = 1: Input is high

---

**NOTE: Writing to read-only registers PxIN**

Writing to these read-only registers results in increased current consumption while the write attempt is active.

---

### 10.2.2 Output Registers (*PxOUT*)

Each bit in each PxOUT register is the value to be output on the corresponding I/O pin when the pin is configured as I/O function, output direction.

- Bit = 0: Output is low
- Bit = 1: Output is high

If the pin is configured as I/O function, input direction and the pullup or pulldown resistor are enabled; the corresponding bit in the PxOUT register selects pullup or pulldown.

- Bit = 0: Pin is pulled down
- Bit = 1: Pin is pulled up

### 10.2.3 Direction Registers (*PxDIR*)

Each bit in each PxDIR register selects the direction of the corresponding I/O pin when it is configured for I/O function. PxDIR register also in most of the cases controls the direction of the I/O when it is configured for peripheral functions. PxDIR bits for I/O pins that are selected for peripheral functions must be set as required by the peripheral functions. For certain secondary functions like eUSCI, the I/O direction is controlled by the secondary function itself and not by the PxDIR register. Refer to device-specific data sheet for more details.

- Bit = 0: Port pin is switched to input direction
- Bit = 1: Port pin is switched to output direction

### 10.2.4 Pullup or Pulldown Resistor Enable Registers (*PxREN*)

Each bit in each PxREN register enables or disables the pullup or pulldown resistor of the corresponding I/O pin. The corresponding bit in the PxOUT register selects if the pin contains a pullup or pulldown.

- Bit = 0: Pullup or pulldown resistor disabled
- Bit = 1: Pullup or pulldown resistor enabled

Table 10-1 summarizes the use of PxDIR, PxREN, and PxOUT for proper I/O configuration.

**Table 10-1. I/O Configuration**

PxDIR	PxREN	PxOUT	I/O Configuration
0	0	x	Input
0	1	0	Input with pulldown resistor
0	1	1	Input with pullup resistor
1	x	x	Output

### 10.2.5 Output Drive Strength Selection Registers (PxDS)

There are two type of I/Os available. One with regular drive strength and the other with high drive strength. Most of the I/Os have regular drive strength while some selected I/Os have high drive strength. See device-specific data sheet for the I/Os with high drive strength. PxDS register is used to select the drive strength of the high drive strength I/Os.

- Bit = 0: High drive strength I/Os are configured for regular drive strength
- Bit = 1: High drive strength I/Os are configured for high drive strength

PxDS register does not have any effect on the I/Os with only regular drive strength.

### 10.2.6 Function Select Registers (PxSEL0, PxSEL1)

Port pins are often multiplexed with other peripheral module functions. See the device-specific data sheet to determine pin functions. Each port pin uses two bits to select the pin function – I/O port or one of the three possible peripheral module function. [Table 10-2](#) shows how to select the various module functions. Each PxSEL bit is used to select the pin function – I/O port or peripheral module function.

**Table 10-2. I/O Function Selection**

PxSEL1	PxSEL0	I/O Function
0	0	General purpose I/O is selected
0	1	Primary module function is selected
1	0	Secondary module function is selected
1	1	Tertiary module function is selected

Setting the PxSEL1 or PxSEL0 bits to a module function does not automatically set the pin direction. Other peripheral module functions may require the PxDIR bits to be configured according to the direction needed for the module function. See the pin schematics in the device-specific data sheet.

When a port pin is selected as an input to peripheral modules, the input signal to those peripheral modules is a latched representation of the signal at the device pin. While PxSEL1 and PxSEL0 is other than 00, the internal input signal follows the signal at the pin for all connected modules. However, if PxSEL1 and PxSEL0 = 00, the input to the peripherals maintain the value of the input signal at the device pin before the PxSEL1 and PxSEL0 bits were reset.

Because the PxSEL1 and PxSEL0 bits do not reside in contiguous addresses, changing both bits at the same time is not possible. For example, an application might need to change P1.0 from general purpose I/O to the tertiary module function residing on P1.0. Initially, P1SEL1 = 00h and P1SEL0 = 00h. To change the function, it would be necessary to write both P1SEL1 = 01h and P1SEL0 = 01h. This is not possible without first passing through an intermediate configuration, and this configuration may not be desirable from an application standpoint. The PxSELC complement register can be used to handle such situations. The PxSELC register always reads 0. Each set bit of the PxSELC register complements the corresponding respective bit of the PxSEL1 and PxSEL0 registers. In the example, with P1SEL1 = 00h and P1SEL0 = 00h initially, writing P1SELC = 01h causes P1SEL1 = 01h and P1SEL0 = 01h to be written simultaneously.

### 10.2.7 Port Interrupts

All Px interrupt flags for a particular port are prioritized, with PxIFG.0 being the highest, and combined to source a single interrupt vector. The highest priority enabled interrupt generates a number in the PxIV register. This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled Px interrupts do not affect the PxIV value. The PxIV registers are half-word access only.

Each PxIFG bit is the interrupt flag for its corresponding I/O pin, and the flag is set when the selected input signal edge occurs at the pin. All PxIFG interrupt flags request an interrupt when their corresponding PxIE bit is set. Software can also set each PxIFG flag, providing a way to generate a software-initiated interrupt.

- Bit = 0: No interrupt is pending

- Bit = 1: An interrupt is pending

Only transitions, not static levels, cause interrupts. If any PxIFG flag becomes set during a Px interrupt service routine or after Px interrupt service routine execution is completed, the set PxIFG flag generates another interrupt. This ensures that each transition is acknowledged.

---

**NOTE: PxIFG flags when changing PxOUT, PxDIR, or PxREN**

Writing to PxOUT, PxDIR, or PxREN can result in setting the corresponding PxIFG flags.

---

Any access (read or write) of the PxIV register automatically resets the highest pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt.

For example, assume that P1IFG.0 has the highest priority. If the P1IFG.0 and P1IFG.2 flags are set when the interrupt service routine accesses the P1IV register, P1IFG.0 is reset automatically. After the completion of P1IFG.0 interrupt service routine, the P1IFG.2 generates another interrupt.

#### 10.2.7.1 Interrupt Edge Select Registers (PxIES)

Each PxIES bit selects the interrupt edge for the corresponding I/O pin.

- Bit = 0: Respective PxIFG flag is set on a low-to-high transition
- Bit = 1: Respective PxIFG flag is set on a high-to-low transition

---

**NOTE: Writing to PxIES**

Writing to PxIES for each corresponding I/O can result in setting the corresponding interrupt flags.

PxIES	PxIN	PxIFG
0 → 1	0	May be set
0 → 1	1	Unchanged
1 → 0	0	Unchanged
1 → 0	1	May be set

---

#### 10.2.7.2 Interrupt Enable Registers (PxIE)

Each PxIE bit enables the associated PxIFG interrupt flag.

- Bit = 0: The interrupt is disabled
- Bit = 1: The interrupt is enabled

---

**NOTE:** When the device operates in any power mode other than low-power modes LPM3, LPM4, and LPMx.5, the interrupt capability is available only when the PxSEL1 and PxSEL0 bits are 0 for the corresponding interrupt capable I/Os. The PxIE and PxIES registers need to be programmed appropriately.

When the device operates in any of the low-power modes LPM3, LPM4, or LPMx.5, the wake-up capability is available for PxSEL1, PxSEL0 bit combinations 00, 01, and 10 of the corresponding wake-up capable I/Os. The PxIE register needs to be programmed appropriately for wake-up. The PxIES register configuration is a don't care and wake-up is triggered upon rising or falling edge of the wake-up event.

## 10.3 I/O Configuration

### 10.3.1 Configuration After Reset

After a reset, all port pins are configured as inputs with their module functions disabled. To prevent floating inputs, all port pins, including unused ones (see [Section 10.3.2](#)), should be configured according to the application needs as early as possible during the initialization procedure.

### 10.3.2 Configuration of Unused Port Pins

To prevent a floating input and to reduce power consumption, unused I/O pins should be configured as I/O function, output direction, and left unconnected on the PC board. The value of the PxOUT bit is don't care, because the pin is unconnected. Alternatively, the integrated pullup or pulldown resistor can be enabled by setting the PxREN bit of the unused pin to prevent a floating input.

---

**NOTE:** Configuring port PJ and shared JTAG pins

The application should make sure that port PJ is configured properly to prevent a floating input. Some pins of port PJ are shared with the JTAG TDI and TDO functions and are initialized to the JTAG functionality on reset. Other pins of Port J are initialized to high-impedance inputs by default.

---

### 10.3.3 Configuration for LPM3 and LPM4 Modes

The digital I/O configuration is retained through LPM3/LPM4 modes. The application must configure the I/Os appropriately to avoid floating conditions and optimize current consumption in the low-power modes. It is possible to wake-up from LPM3/LPM4 modes through wake-up capable I/Os when PxSEL1 and PxSEL0 registers are programmed for combinations 00, 01, and 10 for the respective I/Os. It is not possible to wake-up from LPM3/LPM4 modes when the PxSEL1 and PxSEL0 registers combination is 11. This capability enables wake-up from LPM3/LPM4 modes upon digital peripheral input events like UART receive, Timer capture, and DMA external trigger.

The PxIE register bits must be set to 1 for the respective I/Os to enable the LPM3/LPM4 wake-up. The PxIES register bits are don't care which means the wake-up is triggered upon rise edge or fall edge of the wake-up event. The PxIFG bit is set to 1 for the I/O that triggered the wake-up. The wake-up event can be serviced if the port interrupt is enabled at the NVIC module.

### 10.3.4 Configuration for LPM3.5 and LPM4.5 Modes

When the device enters LPM3.5 or LPM4.5 low-power modes of operation, the state of the I/Os is locked and stored by the device through the low-power modes. Upon exit from these low-power modes, this state remains locked, until explicitly unlocked by the application. In LPM3.5 or LPM4.5 modes, the configuration registers of the digital I/Os get reset, however the locked state of the I/Os ensures that the reset values do not impact the I/O operation. In this case, it is the responsibility of the application to re-initialize the configuration registers appropriately before releasing the lock condition of the I/Os.

---

**NOTE:** Refer to the *Power Control Manager (PCM)* chapter for more details on the bits that control the locking of the state of the I/Os.

---

Before entering LPM3.5, or LPM4.5 modes, the following operations are required for the I/Os:

- Set all I/Os to general-purpose I/Os (PxSEL0 = 00h and PxSEL1 = 00h) and configure as needed. Each I/O can be set to input high impedance, input with pulldown, input with pullup, output high, or output low. It is critical that no inputs are left floating in the application; otherwise, excess current may be drawn in the low-power mode.
- Configuring the I/O in this manner ensures that each pin is in a safe condition before entering LPM3.5 or LPM4.5.
- Optionally, configure input interrupt pins for wake-up from low-power modes. To wake the device from low-power modes, a general-purpose I/O port must contain an input port with interrupt and wake-up

capability. Not all inputs with interrupt capability may offer wake-up from low-power modes. See the device-specific data sheet for details on which ports have this feature. To wake up the device, a port pin must be configured properly before entering the low-power modes. Each port should be configured as general-purpose input. Pulldowns or pullups can be applied if required. Setting the PxIES bit of the corresponding register determines the edge transition that wakes up the device. Last, the PxIE for the port must be enabled.

---

**NOTE:** It is not possible to wake up from a port interrupt if the respective port interrupt flag is already asserted. TI recommends clearing the flag before entering LPM3, LPM4, LPM3.5, or LPM4.5. Any pending flags in this case could then be serviced before the low-power mode entry.

This completes the operations required for the I/Os before entering LPM3.5 or LPM4.5.

As described above, during LPM3.5, or LPM4.5 modes, the I/O pin states are held and locked based on the settings before the low-power mode entry. Note that only the pin conditions are retained. All other port configuration register settings such as PxDIR, PxREN, PxOUT, PxIES, and PxIE contents are lost.

Upon exit from LPM3.5 or LPM4.5 modes, all peripheral registers are set to their default conditions but the I/O pins remain locked while the LOCKLPM5 bit in the PCM is set. Keeping the I/O pins locked ensures that all pin conditions remain stable when entering the active mode, regardless of the default I/O register settings.

When back in active mode, the I/O configuration and I/O interrupt configuration such as PxDIR, PxREN, PxOUT, and PxIES should be restored to the values before entering LPM3.5 or LPM4.5. The LOCKLPM5 bit can then be cleared, which releases the I/O pin conditions and I/O interrupt configuration. Any changes to the port configuration registers while LOCKLPM5 is set have no effect on the I/O pins.

After enabling the I/O interrupts by configuring PxIE and port interrupt enable configuration at NVIC, the I/O interrupt that caused the wake-up can be serviced as indicated by the PxIFG flags. These flags can be used directly, or the corresponding PxIV register may be used. Note that the PxIFG flag cannot be cleared until the LOCKLPM5 bit has been cleared.

---

**NOTE:** It is possible that multiple events occurred on various ports. In these cases, multiple PxIFG flags are set, and it cannot be determined which port caused the I/O wake-up.

## 10.4 Digital I/O Registers

[Table 10-3](#) shows the digital I/O registers and each register's address offset. Refer to the device-specific data sheet for the base address of the Digital I/O module.

[Table 10-3](#) shows the registers as both 8-bit and 16-bit versions. Each 8-bit register accesses a single port (Port 1, Port 2, Port 3, and so on). Port A is a 16-bit combination of Port 1 and Port 2. Similarly, Port B is a combination of Port 3 and Port 4, Port C is a combination of Port 5 and Port 6, and so on. This configuration allows the application to access the ports either individually or in combinations of two.

For a generic 16-bit register that is named *ANYREG*, the suffix "*\_L*" (*ANYREG\_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "*\_H*" (*ANYREG\_H*) refers to the upper byte of the register (bits 8 through 15).

**Table 10-3. Digital I/O Registers**

Offset	Acronym	Register Name	Section
0Eh	P1IV	Port 1 Interrupt Vector	<a href="#">Section 10.4.1</a>
1Eh	P2IV	Port 2 Interrupt Vector	<a href="#">Section 10.4.1</a>
2Eh	P3IV	Port 3 Interrupt Vector	<a href="#">Section 10.4.1</a>
3Eh	P4IV	Port 4 Interrupt Vector	<a href="#">Section 10.4.1</a>
4Eh	P5IV	Port 5 Interrupt Vector	<a href="#">Section 10.4.1</a>
5Eh	P6IV	Port 6 Interrupt Vector	<a href="#">Section 10.4.1</a>
6Eh	P7IV	Port 7 Interrupt Vector	<a href="#">Section 10.4.1</a>
7Eh	P8IV	Port 8 Interrupt Vector	<a href="#">Section 10.4.1</a>
8Eh	P9IV	Port 9 Interrupt Vector	<a href="#">Section 10.4.1</a>
9Eh	P10IV	Port 10 Interrupt Vector	<a href="#">Section 10.4.1</a>
00h	P1IN	Port 1 Input	<a href="#">Section 10.4.2</a>
	or PAIN_L		
02h	P1OUT	Port 1 Output	<a href="#">Section 10.4.3</a>
	or PAOUT_L		
04h	P1DIR	Port 1 Direction	<a href="#">Section 10.4.4</a>
	or PADIR_L		
06h	P1REN	Port 1 Resistor Enable	<a href="#">Section 10.4.5</a>
	or PAREN_L		
08h	P1DS	Port 1 Drive Strength	<a href="#">Section 10.4.6</a>
	or PADS_L		
0Ah	P1SEL0	Port 1 Select 0	<a href="#">Section 10.4.7</a>
	or PASEL0_L		
0Ch	P1SEL1	Port 1 Select 1	<a href="#">Section 10.4.8</a>
	or PASEL1_L		
16h	P1SELC	Port 1 Complement Selection	<a href="#">Section 10.4.9</a>
	or PASELC_L		
18h	P1IES	Port 1 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
	or PAIES_L		
1Ah	P1IE	Port 1 Interrupt Enable	<a href="#">Section 10.4.11</a>
	or PAIE_L		
1Ch	P1IFG	Port 1 Interrupt Flag	<a href="#">Section 10.4.12</a>
	or PAIFG_L		

**Table 10-3. Digital I/O Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Name</b>	<b>Section</b>
01h	P2IN	Port 2 Input	<a href="#">Section 10.4.2</a>
	or PAIN_H		
03h	P2OUT	Port 2 Output	<a href="#">Section 10.4.3</a>
	or PAOUT_H		
05h	P2DIR	Port 2 Direction	<a href="#">Section 10.4.4</a>
	or PADIR_H		
07h	P2REN	Port 2 Resistor Enable	<a href="#">Section 10.4.5</a>
	or PAREN_H		
09h	P2DS	Port 2 Drive Strength	<a href="#">Section 10.4.6</a>
	or PADS_H		
0Bh	P2SEL0	Port 2 Select 0	<a href="#">Section 10.4.7</a>
	or PASEL0_H		
0Dh	P2SEL1	Port 2 Select 1	<a href="#">Section 10.4.8</a>
	or PASEL1_H		
17h	P2SELC	Port 2 Complement Selection	<a href="#">Section 10.4.9</a>
	or PASELC_L		
19h	P2IES	Port 2 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
	or PAIES_H		
1Bh	P2IE	Port 2 Interrupt Enable	<a href="#">Section 10.4.11</a>
	or PAIE_H		
1Dh	P2IFG	Port 2 Interrupt Flag	<a href="#">Section 10.4.12</a>
	or PAIFG_H		
20h	P3IN	Port 3 Input	<a href="#">Section 10.4.2</a>
	or PBIN_L		
22h	P3OUT	Port 3 Output	<a href="#">Section 10.4.3</a>
	or PBOUT_L		
24h	P3DIR	Port 3 Direction	<a href="#">Section 10.4.4</a>
	or PBDIR_L		
26h	P3REN	Port 3 Resistor Enable	<a href="#">Section 10.4.5</a>
	or PBREN_L		
28h	P3DS	Port 3 Drive Strength	<a href="#">Section 10.4.6</a>
	or PBDS_L		
2Ah	P3SEL0	Port 3 Select 0	<a href="#">Section 10.4.7</a>
	or PBSEL0_L		
2Ch	P3SEL1	Port 3 Select 1	<a href="#">Section 10.4.8</a>
	or PBSEL1_L		
36h	P3SELC	Port 3 Complement Selection	<a href="#">Section 10.4.9</a>
	or PBSELC_L		
38h	P3IES	Port 3 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
	or PBIES_L		
3Ah	P3IE	Port 3 Interrupt Enable	<a href="#">Section 10.4.11</a>
	or PBIE_L		
3Ch	P3IFG	Port 3 Interrupt Flag	<a href="#">Section 10.4.12</a>
	or PBIFG_L		

**Table 10-3. Digital I/O Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Name</b>	<b>Section</b>
21h	P4IN	Port 4 Input	<a href="#">Section 10.4.2</a>
	or PBIN_H		
23h	P4OUT	Port 4 Output	<a href="#">Section 10.4.3</a>
	or PBOUT_H		
25h	P4DIR	Port 4 Direction	<a href="#">Section 10.4.4</a>
	or PBDIR_H		
27h	P4REN	Port 4 Resistor Enable	<a href="#">Section 10.4.5</a>
	or PBREN_H		
29h	P4DS	Port 4 Drive Strength	<a href="#">Section 10.4.6</a>
	or PBDS_H		
2Bh	P4SEL0	Port 4 Select 0	<a href="#">Section 10.4.7</a>
	or PBSEL0_H		
2Dh	P4SEL1	Port 4 Select 1	<a href="#">Section 10.4.8</a>
	or PBSEL1_H		
37h	P4SELC	Port 4 Complement Selection	<a href="#">Section 10.4.9</a>
	or PBSELC_L		
39h	P4IES	Port 4 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
	or PBIES_H		
3Bh	P4IE	Port 4 Interrupt Enable	<a href="#">Section 10.4.11</a>
	or PBIE_H		
3Dh	P4IFG	Port 4 Interrupt Flag	<a href="#">Section 10.4.12</a>
	or PBIFG_H		
40h	P5IN	Port 5 Input	<a href="#">Section 10.4.2</a>
	or PCIN_L		
42h	P5OUT	Port 5 Output	<a href="#">Section 10.4.3</a>
	or PCOUT_L		
44h	P5DIR	Port 5 Direction	<a href="#">Section 10.4.4</a>
	or PCDIR_L		
46h	P5REN	Port 5 Resistor Enable	<a href="#">Section 10.4.5</a>
	or PCREN_L		
48h	P5DS	Port 5 Drive Strength	<a href="#">Section 10.4.6</a>
	or PCDS_L		
4Ah	P5SEL0	Port 5 Select 0	<a href="#">Section 10.4.7</a>
	or PCSEL0_L		
4Ch	P5SEL1	Port 5 Select 1	<a href="#">Section 10.4.8</a>
	or PCSEL1_L		
56h	P5SELC	Port 5 Complement Selection	<a href="#">Section 10.4.9</a>
	or PCSELC_L		
58h	P5IES	Port 5 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
	or PCIIES_L		
5Ah	P5IE	Port 5 Interrupt Enable	<a href="#">Section 10.4.11</a>
	or PCIE_L		
5Ch	P5IFG	Port 5 Interrupt Flag	<a href="#">Section 10.4.12</a>
	or PCIFG_L		

**Table 10-3. Digital I/O Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Name</b>	<b>Section</b>
41h	P6IN	Port 6 Input	<a href="#">Section 10.4.2</a>
	or PCIN_H		
43h	P6OUT	Port 6 Output	<a href="#">Section 10.4.3</a>
	or PCOUT_H		
45h	P6DIR	Port 6 Direction	<a href="#">Section 10.4.4</a>
	or PCDIR_H		
47h	P6REN	Port 6 Resistor Enable	<a href="#">Section 10.4.5</a>
	or PCREN_H		
49h	P6DS	Port 6 Drive Strength	<a href="#">Section 10.4.6</a>
	or PCDS_H		
4Bh	P6SEL0	Port 6 Select 0	<a href="#">Section 10.4.7</a>
	or PCSEL0_H		
4Dh	P6SEL1	Port 6 Select 1	<a href="#">Section 10.4.8</a>
	or PCSEL1_H		
57h	P6SELC	Port 6 Complement Selection	<a href="#">Section 10.4.9</a>
	or PCSELC_L		
59h	P6IES	Port 6 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
	or PCIES_H		
5Bh	P6IE	Port 6 Interrupt Enable	<a href="#">Section 10.4.11</a>
	or PCIE_H		
5Dh	P6IFG	Port 6 Interrupt Flag	<a href="#">Section 10.4.12</a>
	or PCIFG_H		
60h	P7IN	Port 7 Input	<a href="#">Section 10.4.2</a>
	or PDIN_L		
62h	P7OUT	Port 7 Output	<a href="#">Section 10.4.3</a>
	or PDOUT_L		
64h	P7DIR	Port 7 Direction	<a href="#">Section 10.4.4</a>
	or PDDIR_L		
66h	P7REN	Port 7 Resistor Enable	<a href="#">Section 10.4.5</a>
	or PDREN_L		
68h	P7DS	Port 7 Drive Strength	<a href="#">Section 10.4.6</a>
	or PDDS_L		
6Ah	P7SEL0	Port 7 Select 0	<a href="#">Section 10.4.7</a>
	or PDSEL0_L		
6Ch	P7SEL1	Port 7 Select 1	<a href="#">Section 10.4.8</a>
	or PDSEL1_L		
76h	P7SELC	Port 7 Complement Selection	<a href="#">Section 10.4.9</a>
	or PDSELC_L		
78h	P7IES	Port 7 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
	or PDIIES_L		
7Ah	P7IE	Port 7 Interrupt Enable	<a href="#">Section 10.4.11</a>
	or PDIE_L		
7Ch	P7IFG	Port 7 Interrupt Flag	<a href="#">Section 10.4.12</a>
	or PDIFG_L		

**Table 10-3. Digital I/O Registers (continued)**

Offset	Acronym	Register Name	Section
61h	P8IN	Port 8 Input	<a href="#">Section 10.4.2</a>
	or PDIN_H		
63h	P8OUT	Port 8 Output	<a href="#">Section 10.4.3</a>
	or PDOUT_H		
65h	P8DIR	Port 8 Direction	<a href="#">Section 10.4.4</a>
	or PDDIR_H		
67h	P8REN	Port 8 Resistor Enable	<a href="#">Section 10.4.5</a>
	or PDREN_H		
69h	P8DS	Port 8 Drive Strength	<a href="#">Section 10.4.6</a>
	or PDDS_H		
6Bh	P8SEL0	Port 8 Select 0	<a href="#">Section 10.4.7</a>
	or PDSEL0_H		
6Dh	P8SEL1	Port 8 Select 1	<a href="#">Section 10.4.8</a>
	or PDSEL1_H		
77h	P8SELC	Port 8 Complement Selection	<a href="#">Section 10.4.9</a>
	or PDSELC_L		
79h	P8IES	Port 8 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
	or PDIES_H		
7Bh	P8IE	Port 8 Interrupt Enable	<a href="#">Section 10.4.11</a>
	or PEIE_L		
7Dh	P8IFG	Port 8 Interrupt Flag	<a href="#">Section 10.4.12</a>
	or PEIFG_H		
80h	P9IN	Port 9 Input	<a href="#">Section 10.4.2</a>
	or PEIN_L		
82h	P9OUT	Port 9 Output	<a href="#">Section 10.4.3</a>
	or PEOUT_L		
84h	P9DIR	Port 9 Direction	<a href="#">Section 10.4.4</a>
	or PEDIR_L		
86h	P9REN	Port 9 Resistor Enable	<a href="#">Section 10.4.5</a>
	or PEREN_L		
88h	P9DS	Port 9 Drive Strength	<a href="#">Section 10.4.6</a>
	or PEDS_L		
8Ah	P9SEL0	Port 9 Select 0	<a href="#">Section 10.4.7</a>
	or PESEL0_L		
8Ch	P9SEL1	Port 9 Select 1	<a href="#">Section 10.4.8</a>
	or PESEL1_L		
96h	P9SELC	Port 9 Complement Selection	<a href="#">Section 10.4.9</a>
	or PESELC_L		
98h	P9IES	Port 9 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
	or PEIES_L		
9Ah	P9IE	Port 9 Interrupt Enable	<a href="#">Section 10.4.11</a>
	or PEIE_L		
9Ch	P9IFG	Port 9 Interrupt Flag	<a href="#">Section 10.4.12</a>
	or PEIFG_L		

**Table 10-3. Digital I/O Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Name</b>	<b>Section</b>
81h	P10IN or PEIN_H	Port 10 Input	<a href="#">Section 10.4.2</a>
83h	P10OUT or PEOUT_H	Port 10 Output	<a href="#">Section 10.4.3</a>
85h	P10DIR or PEDIR_H	Port 10 Direction	<a href="#">Section 10.4.4</a>
87h	P10REN or PEREN_H	Port 10 Resistor Enable	<a href="#">Section 10.4.5</a>
89h	P10DS or PEDS_H	Port 10 Drive Strength	<a href="#">Section 10.4.6</a>
8Bh	P10SEL0 or PESEL0_H	Port 10 Select 0	<a href="#">Section 10.4.7</a>
8Dh	P10SEL1 or PESEL1_H	Port 10 Select 1	<a href="#">Section 10.4.8</a>
97h	P10SELC or PESELC_L	Port 10 Complement Selection	<a href="#">Section 10.4.9</a>
99h	P10IES or PEIES_H	Port 10 Interrupt Edge Select	<a href="#">Section 10.4.10</a>
9Bh	P10IE or PEIE_H	Port 10 Interrupt Enable	<a href="#">Section 10.4.11</a>
9Dh	P10IFG or PEIFG_H	Port 10 Interrupt Flag	<a href="#">Section 10.4.12</a>

**Table 10-3. Digital I/O Registers (continued)**

Offset	Acronym	Register Name	Section
00h	PAIN	Port A Input	
00h	PAIN_L		
01h	PAIN_H		
02h	PAOUT	Port A Output	
02h	PAOUT_L		
03h	PAOUT_H		
04h	PADIR	Port A Direction	
04h	PADIR_L		
05h	PADIR_H		
06h	PAREN	Port A Resistor Enable	
06h	PAREN_L		
07h	PAREN_H		
08h	PADS	Port A Drive Strength	
08h	PADS_L		
09h	PADS_H		
0Ah	PASEL0	Port A Select 0	
0Ah	PASEL0_L		
0Bh	PASEL0_H		
0Ch	PASEL1	Port A Select 1	
0Ch	PASEL1_L		
0Dh	PASEL1_H		
16h	PASELC	Port A Complement Select	
16h	PASELC_L		
17h	PASELC_H		
18h	PAIES	Port A Interrupt Edge Select	
18h	PAIES_L		
19h	PAIES_H		
1Ah	PAIE	Port A Interrupt Enable	
1Ah	PAIE_L		
1Bh	PAIE_H		
1Ch	PAIFG	Port A Interrupt Flag	
1Ch	PAIFG_L		
1Dh	PAIFG_H		

**Table 10-3. Digital I/O Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Name</b>	<b>Section</b>
20h	PBIN	Port B Input	
20h	PBIN_L		
21h	PBIN_H		
22h	PBOUT	Port B Output	
22h	PBOUT_L		
23h	PBOUT_H		
24h	PBDIR	Port B Direction	
24h	PBDIR_L		
25h	PBDIR_H		
26h	PBREN	Port B Resistor Enable	
26h	PBREN_L		
27h	PBREN_H		
28h	PBDS	Port B Drive Strength	
28h	PBDS_L		
29h	PBDS_H		
2Ah	PBSEL0	Port B Select 0	
2Ah	PBSEL0_L		
2Bh	PBSEL0_H		
2Ch	PBSEL1	Port B Select 1	
2Ch	PBSEL1_L		
2Dh	PBSEL1_H		
36h	PBSELC	Port B Complement Select	
36h	PBSELC_L		
37h	PBSELC_H		
38h	PBIOS	Port B Interrupt Edge Select	
38h	PBIOS_L		
39h	PBIOS_H		
3Ah	PBIE	Port B Interrupt Enable	
3Ah	PBIE_L		
3Bh	PBIE_H		
3Ch	PBIFG	Port B Interrupt Flag	
3Ch	PBIFG_L		
3Dh	PBIFG_H		

**Table 10-3. Digital I/O Registers (continued)**

Offset	Acronym	Register Name	Section
40h	PCIN	Port C Input	
40h	PCIN_L		
41h	PCIN_H		
42h	PCOUT	Port C Output	
42h	PCOUT_L		
43h	PCOUT_H		
44h	PCDIR	Port C Direction	
44h	PCDIR_L		
45h	PCDIR_H		
46h	PCREN	Port C Resistor Enable	
46h	PCREN_L		
47h	PCREN_H		
48h	PCDS	Port C Drive Strength	
48h	PCDS_L		
49h	PCDS_H		
4Ah	PCSEL0	Port C Select 0	
4Ah	PCSEL0_L		
4Bh	PCSEL0_H		
4Ch	PCSEL1	Port C Select 1	
4Ch	PCSEL1_L		
4Dh	PCSEL1_H		
56h	PCSELC	Port C Complement Select	
56h	PCSELC_L		
57h	PCSELC_H		
58h	PCIES	Port C Interrupt Edge Select	
58h	PCIES_L		
59h	PCIES_H		
5Ah	PCIE	Port C Interrupt Enable	
5Ah	PCIE_L		
5Bh	PCIE_H		
5Ch	PCIFG	Port C Interrupt Flag	
5Ch	PCIFG_L		
5Dh	PCIFG_H		

**Table 10-3. Digital I/O Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Name</b>	<b>Section</b>
60h	PDIN	Port D Input	
60h	PDIN_L		
61h	PDIN_H		
62h	PDOUT	Port D Output	
62h	PDOUT_L		
63h	PDOUT_H		
64h	PDDR	Port D Direction	
64h	PDDR_L		
65h	PDDR_H		
66h	PDREN	Port D Resistor Enable	
66h	PDREN_L		
67h	PDREN_H		
68h	PDDS	Port D Drive Strength	
68h	PDDS_L		
69h	PDDS_H		
6Ah	PDSEL0	Port D Select 0	
6Ah	PDSEL0_L		
6Bh	PDSEL0_H		
6Ch	PDSEL1	Port D Select 1	
6Ch	PDSEL1_L		
6Dh	PDSEL1_H		
76h	PDSELC	Port D Complement Select	
76h	PDSELC_L		
77h	PDSELC_H		
78h	PDIES	Port D Interrupt Edge Select	
78h	PDIES_L		
79h	PDIES_H		
7Ah	PDIE	Port D Interrupt Enable	
7Ah	PDIE_L		
7Bh	PDIE_H		
7Ch	PDIFG	Port D Interrupt Flag	
7Ch	PDIFG_L		
7Dh	PDIFG_H		

**Table 10-3. Digital I/O Registers (continued)**

Offset	Acronym	Register Name	Section
80h	PEIN	Port E Input	
80h	PEIN_L		
81h	PEIN_H		
82h	PEOUT	Port E Output	
82h	PEOUT_L		
83h	PEOUT_H		
84h	PEDIR	Port E Direction	
84h	PEDIR_L		
85h	PEDIR_H		
86h	PEREN	Port E Resistor Enable	
86h	PEREN_L		
87h	PEREN_H		
88h	PEDS	Port E Drive Strength	
88h	PEDS_L		
89h	PEDS_H		
8Ah	PESEL0	Port E Select 0	
8Ah	PESEL0_L		
8Bh	PESEL0_H		
8Ch	PESEL1	Port E Select 1	
8Ch	PESEL1_L		
8Dh	PESEL1_H		
96h	PESELC	Port E Complement Select	
96h	PESELC_L		
97h	PESELC_H		
98h	PEIES	Port E Interrupt Edge Select	
98h	PEIES_L		
99h	PEIES_H		
9Ah	PEIE	Port E Interrupt Enable	
9Ah	PEIE_L		
9Bh	PEIE_H		
9Ch	PEIFG	Port E Interrupt Flag	
9Ch	PEIFG_L		
9Dh	PEIFG_H		

**Table 10-3. Digital I/O Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Name</b>	<b>Section</b>
120h	PJIN	Port J Input	
120h	PJIN_L		Section 10.4.2
121h	PJIN_H		Section 10.4.2
122h	PJOUT	Port J Output	
122h	PJOUT_L		Section 10.4.3
123h	PJOUT_H		Section 10.4.3
124h	PJDIR	Port J Direction	
124h	PJDIR_L		Section 10.4.4
125h	PJDIR_H		Section 10.4.4
126h	PJREN	Port J Resistor Enable	
126h	PJREN_L		Section 10.4.5
127h	PJREN_H		Section 10.4.5
128h	PJDS	Port J Drive Strength	
128h	PJDS_L		Section 10.4.6
129h	PJDS_H		Section 10.4.6
12Ah	PJSEL0	Port J Select 0	
12Ah	PJSEL0_L		Section 10.4.7
12Bh	PJSEL0_H		Section 10.4.7
12Ch	PJSEL1	Port J Select 1	
12Ch	PJSEL1_L		Section 10.4.8
12Dh	PJSEL1_H		Section 10.4.8
136h	PJSELC	Port J Complement Select	
136h	PJSELC_L		Section 10.4.9
137h	PJSELC_H		Section 10.4.9

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 10.4.1 PxIV Register

Port X Interrupt Vector Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10)

**Figure 10-1. PxIV Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
PxIV							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

**Table 10-4. PxIV Register Description**

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Reads return 0h
4-0	PxIV	R	0h	Port x interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Port x.0 interrupt; Interrupt Flag: PxIFG.0; Interrupt Priority: Highest 04h = Interrupt Source: Port x.1 interrupt; Interrupt Flag: PxIFG.1 06h = Interrupt Source: Port x.2 interrupt; Interrupt Flag: PxIFG.2 08h = Interrupt Source: Port x.3 interrupt; Interrupt Flag: PxIFG.3 0Ah = Interrupt Source: Port x.4 interrupt; Interrupt Flag: PxIFG.4 0Ch = Interrupt Source: Port x.5 interrupt; Interrupt Flag: PxIFG.5 0Eh = Interrupt Source: Port x.6 interrupt; Interrupt Flag: PxIFG.6 10b = Interrupt Source: Port x.7 interrupt; Interrupt Flag: PxIFG.7; Interrupt Priority: Lowest

### 10.4.2 PxIN Register

Port X Input Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

**Figure 10-2. PxIN Register**

7	6	5	4	3	2	1	0
PxIN							
r	r	r	r	r	r	r	r

**Table 10-5. PxIN Register Description**

Bit	Field	Type	Reset	Description
7-0	PxIN	R	Undefined	Port X input 0b = Input is low 1b = Input is high

### 10.4.3 PxOUT Register

Port X Output Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

**Figure 10-3. PxOUT Register**

7	6	5	4	3	2	1	0
PxOUT							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 10-6. PxOUT Register Description**

Bit	Field	Type	Reset	Description
7-0	PxOUT	RW	Undefined	Port X output. When I/O configured to output mode: 0b = Output is low. 1b = Output is high. When I/O configured to input mode and pullups/pulldowns enabled: 0b = Pulldown selected 1b = Pullup selected

### 10.4.4 PxDIR Register

Port X Direction Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

**Figure 10-4. PxDIR Register**

7	6	5	4	3	2	1	0
PxDIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 10-7. PxDIR Register Description**

Bit	Field	Type	Reset	Description
7-0	PxDIR	RW	0h	Port X direction. 0b = Port configured as input 1b = Port configured as output

#### 10.4.5 PxREN Register

Port X Pullup or Pulldown Resistor Enable Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

**Figure 10-5. PxREN Register**

7	6	5	4	3	2	1	0
PxREN							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 10-8. PxREN Register Description**

Bit	Field	Type	Reset	Description
7-0	PxREN	RW	0h	Port X pullup or pulldown resistor enable. When the port is configured as an input, setting this bit enables or disables the pullup or pulldown. 0b = Pullup or pulldown disabled 1b = Pullup or pulldown enabled

#### 10.4.6 PxDS Register

Port X Drive Strength Selection Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

**Figure 10-6. PxDS Register**

7	6	5	4	3	2	1	0
PxDS							
rw-0							

**Table 10-9. PxDS Register Description**

Bit	Field	Type	Reset	Description
7-0	PxDS	RW	0h	Port X drive strength selection (for high drive strength I/Os). 0b = High drive strength I/Os configured for regular drive strength. 1b = High drive strength I/Os configured for high drive strength. If high drive strength is not implemented specific I/Os the corresponding bits in this register will always read 0.

#### 10.4.7 PxSEL0 Register

Port X Function Selection Register 0 (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

**Figure 10-7. PxSEL0 Register**

7	6	5	4	3	2	1	0
PxSEL0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 10-10. PxSEL0 Register Description**

Bit	Field	Type	Reset	Description
7-0	PxSEL0	RW	0h	Port function selection. Each bit corresponds to one channel on Port X. The values of each bit position in PxSEL1 and PxSEL0 are combined to specify the function. For example, if P1SEL1.5 = 1 and P1SEL0.5 = 0, then the secondary module function is selected for P1.5. See PxSEL1 for the definition of each value.

#### 10.4.8 PxSEL1 Register

Port X Function Selection Register 1 (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

**Figure 10-8. PxSEL1 Register**

7	6	5	4	3	2	1	0
PxSEL1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 10-11. PxSEL1 Register Description**

Bit	Field	Type	Reset	Description
7-0	PxSEL1	RW	0h	Port function selection. Each bit corresponds to one channel on Port X. The values of each bit position in PxSEL1 and PxSEL0 are combined to specify the function. For example, if P1SEL1.5 = 1 and P1SEL0.5 = 0, then the secondary module function is selected for P1.5. 00b = General-purpose I/O is selected 01b = Primary module function is selected 10b = Secondary module function is selected 11b = Tertiary module function is selected

#### 10.4.9 PxSELC Register

Port X Complement Selection (X = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or J)

**Figure 10-9. PxSELC Register**

7	6	5	4	3	2	1	0
PxSELC							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 10-12. PxSELC Register Description**

Bit	Field	Type	Reset	Description
7-0	PxSELC	RW	0h	Port selection complement. Each bit that is set in PxSELC complements the corresponding respective bit of both the PxSEL1 and PxSEL0 registers; that is, for each bit set in PxSELC, the corresponding bits in both P1xEL1 and PxSEL0 are both changed at the same time. Always reads as 0.

#### 10.4.10 PxIES Register

Port X Interrupt Edge Select Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10)

**Figure 10-10. PxIES Register**

7	6	5	4	3	2	1	0
PxIES							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 10-13. P1IES Register Description**

Bit	Field	Type	Reset	Description
7-0	PxIES	RW	Undefined	Port X interrupt edge select 0b = PxIFG flag is set with a low-to-high transition. 1b = PxIFG flag is set with a high-to-low transition.

### 10.4.11 PxIE Register

Port X Interrupt Enable Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10)

**Figure 10-11. PxIE Register**

7	6	5	4	3	2	1	0
PxIE							
rw-0							

**Table 10-14. PxIE Register Description**

Bit	Field	Type	Reset	Description
7-0	PxIE	RW	0h	Port X interrupt enable 0b = Corresponding port interrupt disabled 1b = Corresponding port interrupt enabled

### 10.4.12 PxIFG Register

Port X Interrupt Flag Register (X = 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10)

**Figure 10-12. PxIFG Register**

7	6	5	4	3	2	1	0
PxIFG							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 10-15. PxIFG Register Description**

Bit	Field	Type	Reset	Description
7-0	PxIFG	RW	0h	Port X interrupt flag 0b = No interrupt is pending. 1b = Interrupt is pending.

## **Port Mapping Controller (PMAP)**

The port mapping controller allows a flexible mapping of digital functions to port pins. This chapter describes the port mapping controller.

Topic	Page
<b>11.1 Port Mapping Controller Introduction.....</b>	<b>522</b>
<b>11.2 Port Mapping Controller Operation .....</b>	<b>522</b>
<b>11.3 PMAP Registers .....</b>	<b>525</b>

## 11.1 Port Mapping Controller Introduction

The port mapping controller allows the flexible and reconfigurable mapping of digital functions to port pins.

The port mapping controller features are:

- Configuration protected by write access key.
- Default mapping provided for each port pin (device-dependent, the device pinout in the device-specific data sheet).
- Mapping can be reconfigured during runtime.
- Each output signal can be mapped to several output pins.

## 11.2 Port Mapping Controller Operation

The port mapping is configured with user software. The setup is discussed in the following sections.

### 11.2.1 Access

To enable write access to any of the port mapping controller registers, the correct key must be written into the PMAPKEYID register. The PMAPKEYID register always reads 096A5h. Writing the key 02D52h grants write access to all port mapping controller registers. Read access is always possible.

If an invalid key is written while write access is granted, any further write accesses are prevented. It is recommended that the application completes mapping configuration by writing an invalid key.

Interrupts should be disabled during the configuration process or the application should take precautions that the execution of an interrupt service routine does not accidentally cause a permanent lock of the port mapping registers; for example, by using the reconfiguration capability (see [Section 11.2.2](#)).

The access status is reflected in the PMAPLOCK bit.

By default, the port mapping controller allows only one configuration after hard reset. A second attempt to enable write access by writing the correct key is ignored, and the registers remain locked. A hard reset is required to disable the permanent lock again. If it is necessary to reconfigure the mapping during runtime, the PMAPRECFG bit must be set during the first write access timeslot. If PMAPRECFG is cleared during later configuration sessions, no more configuration sessions are possible.

---

**NOTE:** Port mapping functions should not be reconfigured while the respective peripheral functions are active. For example, Timer PWM or eUSCI transmit/receive functions should not be reconfigured while these functions are active.

---

### 11.2.2 Mapping

For each port pin, Px.y, on ports providing the mapping functionality, a mapping register, PxMAPy, is available. Setting this register to a certain value maps a module's input and output signals to the respective port pin Px.y. The port pin itself is switched from a general purpose I/O to the selected peripheral/secondary function by setting the corresponding PxSEL.y bit to 1. If the input or the output function of the module is used, it is typically defined by the setting the PxDIR.y bit. If PxDIR.y = 0, the pin is an input, if PxDIR.y = 1, the pin is an output. There are also peripherals (for example, the eUSCI module) that control the direction or even other functions of the pin (for example, open drain), and these options are documented in the mapping table.

With the port mapping functionality the output of a module can be mapped to multiple pins. Also the input of a module can receive inputs from multiple pins. When mapping multiple inputs onto one function, care needs to be taken because the input signals are logically ORed together without applying any priority; therefore, a logic one on any of the inputs results in a logic one at the module. The external input at device pin must be at a logic zero when the pin configuration is changed from peripheral function to I/O function through changing PxSEL0.y and PxSEL1.y bits to 0.

The mapping is device-dependent; see the device-specific data sheet for available functions and specific values. The use of mapping mnemonics to abstract the underlying PxMAPy values is recommended to allow simple portability between different devices. [Table 11-1](#) shows some examples for mapping mnemonics of some common peripherals.

All mappable port pins provide the function PM\_ANALOG (0FFh). Setting the port mapping register PxMAPy to PM\_ANALOG together with PxSEL.y = 1 disables the output driver and the input Schmitt-trigger, to prevent parasitic cross currents when applying analog signals.

**Table 11-1. Examples for Port Mapping Mnemonics and Functions**

PxMAPy Mnemonic	Input Pin Function With PxSEL.y = 1 and PxDIR.y = 0	Output Pin Function With PxSEL.y = 1 and PxDIR.y = 1
PM_NONE	None	DVSS
PM_ACLK	None	ACLK
PM_MCLK	None	MCLK
PM_SMCLK	None	SMCLK
PM_TA0CLK	TimerA0 clock input	DVSS
PM_TA0.0	TimerA0 CCR0 capture input CCI0A	TimerA0 CCR0 compare output Out0
PM_TA0.1	TimerA0 CCR1 capture input CCI1A	TimerA0 CCR1 compare output Out1
PM_TA0.2	TimerA0 CCR2 capture input CCI2A	TimerA0 CCR2 compare output Out2
PM_TA0.3	TimerA0 CCR3 capture input CCI3A	TimerA0 CCR3 compare output Out3
PM_TA0.4	TimerA0 CCR4 capture input CCI4A	TimerA0 CCR4 compare output Out4
PM_TA1CLK	TimerA1 clock input	DVSS
PM_TA1.0	TimerA1 CCR0 capture input CCI0A	TimerA1 CCR0 compare output Out0
PM_TA1.1	TimerA1 CCR1 capture input CCI1A	TimerA1 CCR1 compare output Out1
PM_TA1.2	TimerA1 CCR2 capture input CCI2A	TimerA1 CCR2 compare output Out2
PM_TB0CLK	TimerB0 clock input	DVSS
PM_TB0OUTH	TimerB0 outputs high impedance	DVSS
PM_TB0.0	TimerB0 CCR0 capture input CCI0A	TimerB0 CCR0 compare output Out0 [direction controlled by Timer_B (TBOUTH)]
PM_TB0.1	TimerB0 CCR1 capture input CCI1A	TimerB0 CCR1 compare output Out1 [direction controlled by Timer_B (TBOUTH)]
PM_TB0.2	TimerB0 CCR2 capture input CCI2A	TimerB0 CCR2 compare output Out2 [direction controlled by Timer_B (TBOUTH)]
PM_TB0.3	TimerB0 CCR3 capture input CCI3A	TimerB0 CCR3 compare output Out3 [direction controlled by Timer_B (TBOUTH)]
PM_TB0.4	TimerB0 CCR4 capture input CCI4A	TimerB0 CCR4 compare output Out4 [direction controlled by Timer_B (TBOUTH)]
PM_TB0.5	TimerB0 CCR5 capture input CCI5A	TimerB0 CCR5 compare output Out5 [direction controlled by Timer_B (TBOUTH)]
PM_TB0.6	TimerB0 CCR6 capture input CCI6A	TimerB0 CCR6 compare output Out6 [direction controlled by Timer_B (TBOUTH)]
PM_UCA0RXD	eUSCI_A0 UART RXD (direction controlled by eUSCI - input)	
PM_UCA0SOMI	eUSCI_A0 SPI slave out master in (direction controlled by eUSCI)	
PM_UCA0TXD	eUSCI_A0 UART TXD (direction controlled by eUSCI - output)	
PM_UCA0SIMO	eUSCI_A0 SPI slave in master out (direction controlled by eUSCI)	
PM_UCA0CLK	eUSCI_A0 clock input/output (direction controlled by eUSCI)	
PM_UCA0STE	eUSCI_A0 SPI slave transmit enable (direction controlled by eUSCI)	
PM_UCB0SOMI	eUSCI_B0 SPI slave out master in (direction controlled by eUSCI)	
PM_UCB0SCL	eUSCI_B0 I2C clock (open drain and direction controlled by eUSCI)	
PM_UCB0SIMO	eUSCI_B0 SPI slave in master out (direction controlled by eUSCI)	
PM_UCB0SDA	eUSCI_B0 I2C data (open drain and direction controlled by eUSCI)	
PM_UCB0CLK	eUSCI_B0 clock input/output (direction controlled by eUSCI)	
PM_UCB0STE	eUSCI_B0 SPI slave transmit enable (direction controlled by eUSCI)	

**Table 11-1. Examples for Port Mapping Mnemonics and Functions (continued)**

PxMAPy Mnemonic	Input Pin Function With PxSEL.y = 1 and PxDIR.y = 0	Output Pin Function With PxSEL.y = 1 and PxDIR.y = 1
PM_ANALOG	Disables the output driver and the input Schmitt-trigger to prevent parasitic cross currents when applying analog signals	

## 11.3 PMAP Registers

The registers for the port mapping controller are listed in [Table 11-2](#). The mapping registers can be accessed as bytes or as half-words as shown in this table. The base address of the port mapping controller can be found in the device-specific data sheet.

**Table 11-2. PMAP Registers**

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	PMAPKEYID	Port mapping key register	Read/Write	Half-word	96A5h	<a href="#">Section 11.3.1</a>
02h	PMAPCTL	Port mapping control register	Read/Write	Half-word	0001h	<a href="#">Section 11.3.2</a>
<b>Byte Access Registers</b>						
08h to 0Fh	P1MAP0 to P1MAP7	Port mapping register P1.0 to Port mapping register P1.7	Read/write	Byte	X	<a href="#">Section 11.3.3</a>
10h to 17h	P2MAP0 to P2MAP7	Port mapping register P2.0 to Port mapping register P2.7	Read/write	Byte	X	<a href="#">Section 11.3.4</a>
18h to 1Fh	P3MAP0 to P3MAP7	Port mapping register P3.0 to Port mapping register P3.7	Read/write	Byte	X	<a href="#">Section 11.3.5</a>
20h to 27h	P4MAP0 to P4MAP7	Port mapping register P4.0 to Port mapping register P4.7	Read/write	Byte	X	<a href="#">Section 11.3.6</a>
28h to 2Fh	P5MAP0 to P5MAP7	Port mapping register P5.0 to Port mapping register P5.7	Read/write	Byte	X	<a href="#">Section 11.3.7</a>
30h to 37h	P6MAP0 to P6MAP7	Port mapping register P6.0 to Port mapping register P6.7	Read/write	Byte	X	<a href="#">Section 11.3.8</a>
38h to 3Fh	P7MAP0 to P7MAP7	Port mapping register P7.0 to Port mapping register P7.7	Read/write	Byte	X	<a href="#">Section 11.3.9</a>
<b>Half-Word Access Registers</b>						
08h	P1MAP01	Port mapping register, P1.0 and P1.1	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
0Ah	P1MAP23	Port mapping register, P1.2 and P1.3	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
0Ch	P1MAP45	Port mapping register, P1.4 and P1.5	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
0Eh	P1MAP67	Port mapping register, P1.6 and P1.7	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
10h	P2MAP01	Port mapping register, P2.0 and P2.1	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
12h	P2MAP23	Port mapping register, P2.2 and P2.3	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
14h	P2MAP45	Port mapping register, P2.4 and P2.5	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
16h	P2MAP67	Port mapping register, P2.6 and P2.7	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
18h	P3MAP01	Port mapping register, P3.0 and P3.1	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
1Ah	P3MAP23	Port mapping register, P3.2 and P3.3	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
1Ch	P3MAP45	Port mapping register, P3.4 and P3.5	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
1Eh	P3MAP67	Port mapping register, P3.6 and P3.7	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
20h	P4MAP01	Port mapping register, P4.0 and P4.1	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
22h	P4MAP23	Port mapping register, P4.2 and P4.3	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
24h	P4MAP45	Port mapping register, P4.4 and P4.5	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
26h	P4MAP67	Port mapping register, P4.6 and P4.7	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
28h	P5MAP01	Port mapping register, P5.0 and P5.1	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
2Ah	P5MAP23	Port mapping register, P5.2 and P5.3	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
2Ch	P5MAP45	Port mapping register, P5.4 and P5.5	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
2Eh	P5MAP67	Port mapping register, P5.6 and P5.7	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
30h	P6MAP01	Port mapping register, P6.0 and P6.1	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
32h	P6MAP23	Port mapping register, P6.2 and P6.3	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
34h	P6MAP45	Port mapping register, P6.4 and P6.5	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
36h	P6MAP67	Port mapping register, P6.6 and P6.7	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
38h	P7MAP01	Port mapping register, P7.0 and P7.1	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
3Ah	P7MAP23	Port mapping register, P7.2 and P7.3	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>

**Table 11-2. PMAP Registers (continued)**

Offset	Acronym	Register Name	Type	Access	Reset	Section
3Ch	P7MAP45	Port mapping register, P7.4 and P7.5	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>
3Eh	P7MAP67	Port mapping register, P7.6 and P7.7	Read/write	Half-word	X	<a href="#">Section 11.3.10</a>

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 11.3.1 PMAPKEYID Register (offset = 00h) [reset = 96A5h]

Port Mapping Key Register

**Figure 11-1. PMAPKEYID Register**

15	14	13	12	11	10	9	8
PMAPKEYx							
7	6	5	4	3	2	1	0
PMAPKEYx							

**Table 11-3. PMAPKEYID Register Description**

Bit	Field	Type	Reset	Description
15-0	PMAPKEYx	RW	96A5h	Port mapping controller write access key. Always reads 096A5h. Must be written 02D52h for write access to the port mapping registers.

### 11.3.2 PMAPCTL Register (offset = 02h) [reset = 0001h]

Port Mapping Control Register

**Figure 11-2. PMAPCTL Register**

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved						PMAPRECFG	PMAPLOCKED
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	r-1

**Table 11-4. PMAPCTL Register Description**

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved. Always reads as 0.
1	PMAPRECFG	RW	0h	Port mapping reconfiguration control bit 0b = Configuration allowed only once 1b = Allow reconfiguration of port mapping
0	PMAPLOCKED	R	1h	Port mapping lock bit. Read only 0b = Access to mapping registers is granted 1b = Access to mapping registers is locked

### 11.3.3 P1MAP0 to P1MAP7 Register (offset = 08h to 0Fh) [reset = X]

Port Mapping Register, P1.x (x = 0 to 7)

**Figure 11-3. P1MAP0 to P1MAP7 Register**

7	6	5	4	3	2	1	0
PMAPx							
rw <sup>(1)</sup>							

<sup>(1)</sup> If not all bits are required to decode all provided functions, the unused bits are r-0.

**Table 11-5. P1MAP0 to P1MAP7 Register Description**

Bit	Field	Type	Reset	Description
7-0	PMAPx	RW	X	Selects secondary port function for P1.x. Settings and reset value are device-dependent; see the device-specific data sheet.

#### 11.3.4 P2MAP0 to P2MAP7 Register (offset = 10h to 17h) [reset = X]

Port Mapping Register, P2.x (x = 0 to 7)

**Figure 11-4. P2MAP0 to P2MAP7 Register**

7	6	5	4	3	2	1	0
PMAPx							
rw <sup>(1)</sup>							

<sup>(1)</sup> If not all bits are required to decode all provided functions, the unused bits are r-0.

**Table 11-6. P2MAP0 to P2MAP7 Register Description**

Bit	Field	Type	Reset	Description
7-0	PMAPx	RW	X	Selects secondary port function for P2.x. Settings and reset value are device-dependent; see the device-specific data sheet.

#### 11.3.5 P3MAP0 to P3MAP7 Register (offset = 18h to 1Fh) [reset = X]

Port Mapping Register, P3.x (x = 0 to 7)

**Figure 11-5. P3MAP0 to P3MAP7 Register**

7	6	5	4	3	2	1	0
PMAPx							
rw <sup>(1)</sup>							

<sup>(1)</sup> If not all bits are required to decode all provided functions, the unused bits are r-0.

**Table 11-7. P3MAP0 to P3MAP7 Register Description**

Bit	Field	Type	Reset	Description
7-0	PMAPx	RW	X	Selects secondary port function for P3.x. Settings and reset value are device-dependent; see the device-specific data sheet.

#### 11.3.6 P4MAP0 to P4MAP7 Register (offset = 20h to 27h) [reset = X]

Port Mapping Register, P4.x (x = 0 to 7)

**Figure 11-6. P4MAP0 to P4MAP7 Register**

7	6	5	4	3	2	1	0
PMAPx							
rw <sup>(1)</sup>							

<sup>(1)</sup> If not all bits are required to decode all provided functions, the unused bits are r-0.

**Table 11-8. P4MAP0 to P4MAP7 Register Description**

Bit	Field	Type	Reset	Description
7-0	PMAPx	RW	X	Selects secondary port function for P4.x. Settings and reset value are device-dependent; see the device-specific data sheet.

#### 11.3.7 P5MAP0 to P5MAP7 Register (offset = 28h to 2Fh) [reset = X]

Port Mapping Register, P5.x (x = 0 to 7)

**Figure 11-7. P5MAP0 to P5MAP7 Register**

7	6	5	4	3	2	1	0
PMAPx							
rw <sup>(1)</sup>							

<sup>(1)</sup> If not all bits are required to decode all provided functions, the unused bits are r-0.

**Table 11-9. P5MAP0 to P5MAP7 Register Description**

Bit	Field	Type	Reset	Description
7-0	PMAPx	RW	X	Selects secondary port function for P5.x. Settings and reset value are device-dependent; see the device-specific data sheet.

### 11.3.8 P6MAP0 to P6MAP7 Register (offset = 30h to 37h) [reset = X]

Port Mapping Register, P6.x (x = 0 to 7)

**Figure 11-8. P6MAP0 to P6MAP7 Register**

7	6	5	4	3	2	1	0
PMAPx							
rw <sup>(1)</sup>							

<sup>(1)</sup> If not all bits are required to decode all provided functions, the unused bits are r-0.

**Table 11-10. P6MAP0 to P6MAP7 Register Description**

Bit	Field	Type	Reset	Description
7-0	PMAPx	RW	X	Selects secondary port function for P6.x. Settings and reset value are device-dependent; see the device-specific data sheet.

### 11.3.9 P7MAP0 to P7MAP7 Register (offset = 38h to 3Fh) [reset = X]

Port Mapping Register, P7.x (x = 0 to 7)

**Figure 11-9. P7MAP0 to P7MAP7 Register**

7	6	5	4	3	2	1	0
PMAPx							
rw <sup>(1)</sup>							

<sup>(1)</sup> If not all bits are required to decode all provided functions, the unused bits are r-0.

**Table 11-11. P7MAP0 to P7MAP7 Register Description**

Bit	Field	Type	Reset	Description
7-0	PMAPx	RW	X	Selects secondary port function for P7.x. Settings and reset value are device-dependent; see the device-specific data sheet.

### 11.3.10 PxMAPyz Register [reset = X]

Port Mapping Register Px.y and Px.z (x = 1 to 7, y = [0, 2, 4, 6], z = y + 1)

**Figure 11-10. PxMAPyz Register**

15	14	13	12	11	10	9	8
PMAPx							
rw <sup>(1)</sup>							
PMAPx							
rw <sup>(1)</sup>							

<sup>(1)</sup> If not all bits are required to decode all provided functions, the unused bits are r-0.

**Table 11-12. PxMAPyz Register Description**

Bit	Field	Type	Reset	Description
15-0	PMAPx	RW	undef	Selects secondary port function. Settings and reset value are device-dependent; see the device-specific data sheet.

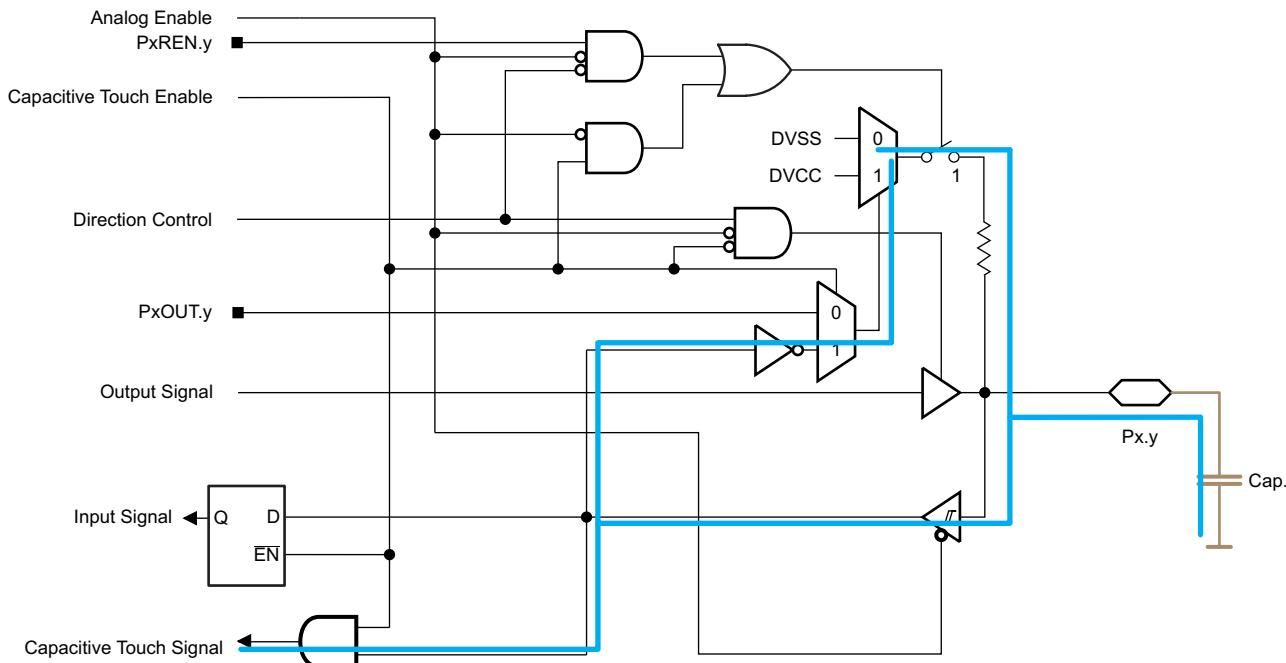
## **Capacitive Touch IO (CAPTIO)**

This chapter describes the functionality of the Capacitive Touch IOs and related control.

Topic	Page
<b>12.1 Capacitive Touch IO Introduction .....</b>	<b>531</b>
<b>12.2 Capacitive Touch IO Operation.....</b>	<b>532</b>
<b>12.3 CapTouch Registers .....</b>	<b>533</b>

## 12.1 Capacitive Touch IO Introduction

The Capacitive Touch IO module allows implementation of a simple capacitive touch sense application. The module uses the integrated pullup and pulldown resistors and an external capacitor to form an oscillator by feeding back the inverted input voltage sensed by the input Schmitt triggers to the pullup and pulldown control. [Figure 12-1](#) shows the capacitive touch IO principle



**Figure 12-1. Capacitive Touch IO Principle**

Figure 12-2 shows the block diagram of the Capacitive Touch IO module.

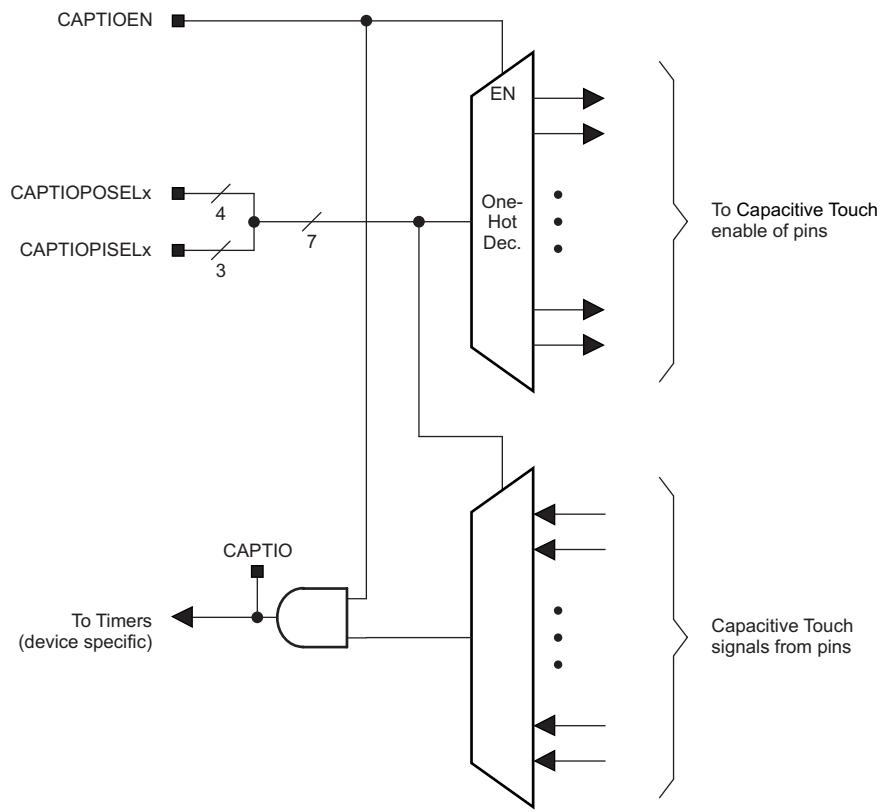


Figure 12-2. Capacitive Touch IO Block Diagram

## 12.2 Capacitive Touch IO Operation

Enable the Capacitive Touch IO functionality with  $\text{CAPTIOEN} = 1$  and select a port pin using  $\text{CAPTIPOSELx}$  and  $\text{CAPTIPISELx}$ . The selected port pin is switched into the Capacitive Touch state, and the resulting oscillating signal is provided to be measured by a timer. The connected timers are device-specific (see the device-specific data sheet).

It is possible to scan to successive port pins by incrementing the low byte of the Capacitive Touch IO control register  $\text{CAPTIOCTL\_L}$  by 2.

## 12.3 CapTouch Registers

The Capacitive Touch IO registers and their address offsets are listed in [Table 12-1](#). In a given device, multiple Capacitive Touch IO registers might be available. The base address of each Capacitive Touch IO module can be found in the device-specific data sheet.

---

**NOTE:** All registers have half-word or byte access. For a generic register ANYREG, the suffix "\_L" (ANYREG\_L) refers to the lower byte of the register (bits 0 through 7). The suffix "\_H" (ANYREG\_H) refers to the upper byte of the register (bits 8 through 15).

---

**Table 12-1. CapTouch Registers**

Offset	Acronym	Register Name	Type	Access	Reset	Section
0Eh	CAPTIOxCTL	Capacitive Touch IO x control register	Read/write	Half-word	0000h	<a href="#">Section 12.3.1</a>
0Eh	CAPTIOxCTL_L		Read/write	Byte	00h	
0Fh	CAPTIOxCTL_H		Read/write	Byte	00h	

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

**12.3.1 CAPTIOxCTL Register (offset = 0Eh) [reset = 0000h]**

Capacitive Touch IO x Control Register

**Figure 12-3. CAPTIOxCTL Register**

15	14	13	12	11	10	9	8
Reserved				CAPTIO	CAPTIOEN		
r0	r0	r0	r0	r0	r0	r-0	rw-0
7	6	5	4	3	2	1	0
CAPTIOPOSELx				CAPTIOPISELx			Reserved
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r0

**Table 12-2. CAPTIOxCTL Register Description**

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved. Always reads 0.
9	CAPTIO	R	0h	Capacitive Touch IO state. Reports the current state of the selected Capacitive Touch IO. Reads 0, if Capacitive Touch IO disabled. 0b = Current state 0 or Capacitive Touch IO is disabled 1b = Current state 1
8	CAPTIOEN	RW	0h	Capacitive Touch IO enable 0b = All Capacitive Touch IOs are disabled. Signal towards timers is 0. 1b = Selected Capacitive Touch IO is enabled
7-4	CAPTIOPOSELx	RW	0h	Capacitive Touch IO port select. Selects port Px. Selecting a port pin that is not available on the device in use gives unpredictable results. 0000b = Px = PJ 0001b = Px = P1 0010b = Px = P2 0011b = Px = P3 0100b = Px = P4 0101b = Px = P5 0110b = Px = P6 0111b = Px = P7 1000b = Px = P8 1001b = Px = P9 1010b = Px = P10 1011b = Px = P11 1100b = Px = P12 1101b = Px = P13 1110b = Px = P14 1111b = Px = P15
3-1	CAPTIOPISELx	RW	0h	Capacitive Touch IO pin select. Selects the pin within selected port Px (see CAPTIOPOSELx). Selecting a port pin that is not available on the device in use gives unpredictable results. 000b = Px.0 001b = Px.1 010b = Px.2 011b = Px.3 100b = Px.4 101b = Px.5 110b = Px.6 111b = Px.7
0	Reserved	R	0h	Reserved. Always reads 0.

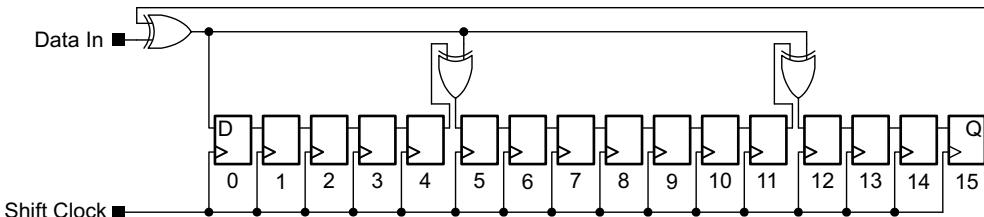
## **CRC32 Module**

The 16-bit or 32-bit cyclic redundancy check (CRC32) module provides a signature for a given data sequence. This chapter describes the operation and use of the CRC32 module.

Topic	Page
13.1 Cyclic Redundancy Check (CRC32) Module Introduction .....	536
13.2 CRC Checksum Generation .....	536
13.3 CRC32 Registers .....	538

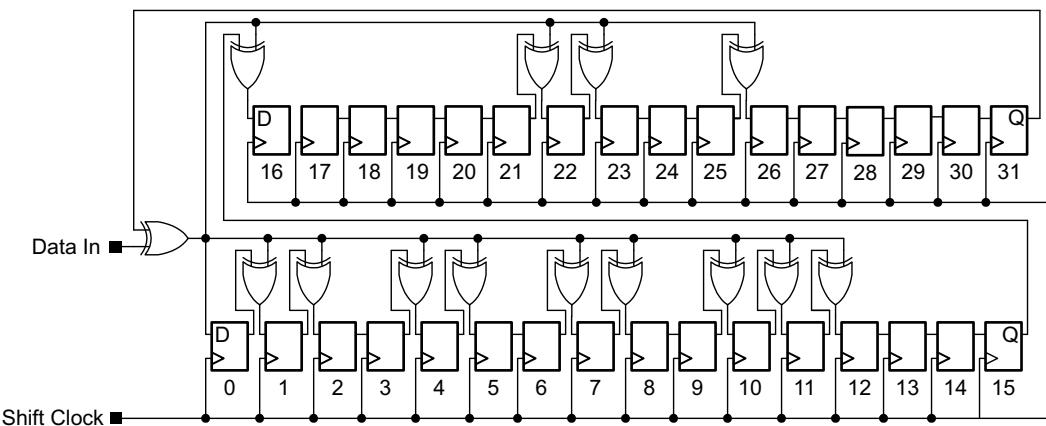
### 13.1 Cyclic Redundancy Check (CRC32) Module Introduction

The CRC module produces a signature for a given sequence of data values. These signatures are defined bit serial in various standard specifications. For CRC16-CCITT, a feedback path from data bits 4, 11, and 15 is generated (see Figure 13-1). This CRC signature is based on the polynomial given in the CRC-CCITT with  $f(x) = x^{16} + x^{12} + x^5 + 1$ .



**Figure 13-1. LFSR Implementation of CRC-CCITT as Defined in Standard (Bit0 is MSB)**

For CRC32-IS3309, a feedback path from data bits 0, 1, 3, 4, 6, 7, 9, 10, 11, 15, 21, 22, 26, and 31 is generated (see Figure 13-2). This CRC signature is based on the polynomial given in the CRC32-ISO3309 with  $f(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ .



**Figure 13-2. LFSR Implementation of CRC32-ISO3309 as Defined in Standard (Bit0 is MSB)**

Identical input data sequences result in identical signatures when the CRC is initialized with a fixed seed value, whereas different sequences of input data, in general, result in different signatures for a given CRC function.

The CRC32 module supports 16-bit and 32-bit CRC generation independent from each other (supported by two dedicated register sets).

### 13.2 CRC Checksum Generation

The CRC generators are first initialized by writing the seed to the CRC Initialization and Result (CRC16INIRES or CRC32INIRES) registers. Any data that should be included into the CRC calculation must be written to the CRC Data Input (CRC16DI or CRC32DI) registers in the same order that the original CRC signatures were calculated. The actual signature can be read from the CRC16INIRES or CRC32INIRES registers to compare the computed checksum with the expected checksum. Signature generation describes a method on how the result of a signature operation can be calculated. The calculated signature, which is computed by an external tool, is called checksum in this chapter. The checksum is stored in the MCU memory and is used to check the correctness of the CRC operation result.

### 13.2.1 CRC Standard and Bit Order

The definitions of the various CRC standards were done in the era of main frame computers. At that time, Bit 0 was treated as MSB. Now, Bit 0 typically denotes LSB (...as value of Bit  $N = 2N$ ). In [Figure 13-1](#) and [Figure 13-2](#) the bit references are used as given in the original standards. The MSP432 MCUs treat Bit 0 as LSB, as is typical in modern CPUs and MCUs.

The fact that Bit 0 is treated as LSB by some and as MSB by others can lead to confusion. The CRC32 module, therefore, provides a bit-reversed register pair for CRC16 and CRC32 operations to support both conventions.

### 13.2.2 CRC Implementation

To allow faster processing of the CRC, the linear feedback shift register (LFSR) functionality is implemented with a set of XOR trees. This implementation shows the identical behavior as the LFSR approach. After a set of 8, 16, or 32 bits is provided to the CRC32 module by writing to the CRC16DI or CRC32DI registers, a calculation for the whole set of input bits is performed. Either the CPU or the DMA can write to the memory mapped data input registers. After the last value is written to CRC16DI or CRC32DIRB, the signature can be read from the CRC16INIRES or CRC32INIRES\_LO and CRC32INIRES\_HI registers after a delay of one clock cycle (required to complete the computation for the last written value). The CRC16 and CRC32 generators accept byte and 16-bit accesses to the input registers CRC16DI and CRC32DI.

For bit-reversed conventions, write the data bytes to the CRC16DIRB or CRC32DIRB registers.

Initialization is done by writing to CRC, and CRC engine adds them to the signature. The bits among each byte are reversed. Data bytes written to CRCDI in 16 bit mode or the data byte in byte mode are not bit reversed before use by the CRC engine. If the checksum itself (with reversed bit order) is included in the CRC operation (as data written to CRCDI or CRCDIRB), the result in the CRCINIRES and CRCRESR registers must be zero.

### 13.3 CRC32 Registers

#### CRC32 Registers

**Table 13-1** lists the offset addresses for the registers in the CRC32 module. Refer to the device-specific data sheet for the base address of the CRC32 module. Any offset not listed in this table should be treated as reserved.

**Table 13-1. CRC32 Registers**

Offset	Acronym	Register Name	Section
0000h	CRC32DI	CRC32 Data Input Low	<a href="#">Section 13.3.0.1</a>
0002h	Reserved	Reserved	
0004h	CRC32DIRB	CRC32 Data In Reverse Low	<a href="#">Section 13.3.0.2</a>
0006h	Reserved	Reserved	
0008h	CRC32INIRES_LO	CRC32 Initialization and Result Low	<a href="#">Section 13.3.0.3</a>
000Ah	CRC32INIRES_HI	CRC32 Initialization and Result High	<a href="#">Section 13.3.0.4</a>
000Ch	CRC32RESR_LO	CRC32 Result Reverse Low	<a href="#">Section 13.3.0.5</a>
000Eh	CRC32RESR_HI	CRC32 Result Reverse High	<a href="#">Section 13.3.0.6</a>
0010h	CRC16DI	CRC16 Data Input Low	<a href="#">Section 13.3.0.7</a>
0012h	Reserved	Reserved	
0014h	CRC16DIRB	CRC16 Data In Reverse Low	<a href="#">Section 13.3.0.8</a>
0016h	Reserved	Reserved	
0018h	CRC16INIRES	CRC16 Init and Result	<a href="#">Section 13.3.0.9</a>
001Ah	Reserved	Reserved	
001Ch	Reserved	Reserved	
001Eh	CRC16RESR	CRC16 Result Reverse	<a href="#">Section 13.3.0.10</a>

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 13.3.0.1 CRC32DI Register

Data Input for CRC32 Signature Computation

**Figure 13-3. CRC32DI Register**

15	14	13	12	11	10	9	8
CRC32DI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
CRC32DI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 13-2. CRC32DI Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC32DI	RW	0h	Data input for CRC32 Computation. Data written to the register is included to the present signature in the CRC32INIRES_HI and CRC32INIRES_LO registers according to the CRC32-ISO3309 standard.

### 13.3.0.2 CRC32DIRB Register

Data In Reverse for CRC32 Computation

**Figure 13-4. CRC32DIRB Register**

15	14	13	12	11	10	9	8
CRC32DIRB							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
CRC32DIRB							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 13-3. CRC32DIRB Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC32DIRB	RW	0h	CRC32 data in bit reversed. Data written to the register is included to the present signature in the CRC32INIRES_HI and CRC32INIRES_LO registers according to the CRC32-ISO3309 standard.

### **13.3.0.3 CRC32INIRES\_LO Register**

CRC32 Initialization and Result, Lower 16 Bits

Data written to this register represents the seed for the CRC calculation. This register always reflects the latest signature of the values collected so far.

**Figure 13-5. CRC32INIRES\_LO Register**

15	14	13	12	11	10	9	8
CRC32INIRES_LO							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
CRC32INIRES_LO							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 13-4. CRC32INIRES\_LO Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC32INIRES_LO <sup>(1)</sup>	RW	0h	CRC32 initialization and result, lower 16 bits. This register holds lower 16 bits of the current CRC32 result (according to the CRC32-ISO3309 standard). Writing to this register initializes the CRC32 calculation with the value written to it.

<sup>(1)</sup> This register is updated with the final signature value one cycle **after** the last data input value is written to the CRC32DI or CRC32DIRB registers. Application should wait for this one cycle delay before reading the result.

### 13.3.0.4 CRC32INIRES\_HI Register

CRC32 Initialization and Result, Upper 16 Bits

Data written to this register represents the seed for the CRC calculation. This register always reflects the latest signature of the values collected so far.

**Figure 13-6. CRC32INIRES\_HI Register**

15	14	13	12	11	10	9	8
CRC32INIRES_HI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC32INIRES_HI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 13-5. CRC32INIRES\_HI Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC32INIRES_HI <sup>(1)</sup>	RW	0h	CRC32 initialization and result, upper 16 bits. This register holds the upper 16 bits of the current CRC32 result (according to the CRC32-ISO3309 standard). Writing to this register initializes the CRC32 calculation with the value written to it.

<sup>(1)</sup> This register is updated with the final signature value one cycle **after** the last data input value is written to the CRC32DI or CRC32DIRB registers. Application should wait for this one cycle delay before reading the result.

### **13.3.0.5 CRC32RESR\_LO Register**

CRC32 Result Reverse, Lower 16 Bits

This register always reflects the latest signature of the values collected so far.

**Figure 13-7. CRC32RESR\_LO Register**

15	14	13	12	11	10	9	8
CRC32RESR_LO							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
CRC32RESR_LO							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 13-6. CRC32RESR\_LO Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC32RESR_LO <sup>(1)</sup>	RW	FFFFh	CRC32 reverse result, lower 16 bits. This register holds the lower 16 bits of the current CRC32 result (according to the CRC32-ISO3309 standard). The order of bits is reversed to the order of bits in the CRC32INIRES register. The entire 32bit result is reversed and hence both upper and lower result registers must be factored in.  For example, CRC32RESR_LO[0] = CRC32INIRES_HI[15], CRC32RESR_LO[10] = CRC32INIRES_HI[5] and so on.

<sup>(1)</sup> This register is updated with the final signature value one cycle **after** the last data input value is written to the CRC32DI or CRC32DIRB registers. Application should wait for this one cycle delay before reading the result.

### 13.3.0.6 CRC32RESR\_HI Register

CRC32 Result Reverse, Upper 16 Bits

This register always reflects the latest signature of the values collected so far.

**Figure 13-8. CRC32RESR\_HI Register**

15	14	13	12	11	10	9	8
CRC32RESR_HI							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
CRC32RESR_HI							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 13-7. CRC32RESR\_HI Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC32RESR_HI <sup>(1)</sup>	RW	FFFFh	CRC32 reverse result, upper 16 bits. This register holds the upper 16 bits of the current CRC32 result (according to the CRC32-ISO3309 standard). The order of bits is reversed to the order of bits in the CRC32INIRES register. The entire 32bit result is reversed and hence both upper and lower result registers must be factored in.  For example, CRC32RESR_HI[0] = CRC32INIRES_LO[15], CRC32RESR_HI[10] = CRC32INIRES_LO[5] and so on.

- <sup>(1)</sup> This register is updated with the final signature value one cycle **after** the last data input value is written to the CRC32DI or CRC32DIRB registers. Application should wait for this one cycle delay before reading the result.

### 13.3.0.7 CRC16DI Register

Data Input for CRC16 Computation

Data written to this register is used in CRC16 signature calculations.

**Figure 13-9. CRC16DI Register**

15	14	13	12	11	10	9	8
CRC16DI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CRC16DI							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 13-8. CRC16DI Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC16DI	RW	0h	CRC16 data in. Data written to the CRC16DI register is included to the present signature in the CRC16INIRES register according to the CRC16-CCITT standard.

### 13.3.0.8 CRC16DIRB Register

CRC16 Data In Reverse Register

Data written to this register is used in CRC16 signature calculations

**Figure 13-10. CRC16DIRB Register**

15	14	13	12	11	10	9	8
CRC16DIRB							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
CRC16DIRB							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 13-9. CRC16DIRB Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC16DIRB	RW	0h	CRC16 data in reverse byte. Data written to the CRC16DIRB register is included to the present signature in the CRC16INIRES and CRC16RESR registers according to the CRC-CCITT standard. Reading the register returns the register CRC16DI content.

### 13.3.0.9 CRC16INIRES Register

CRC16 Initialization and Result Register

**Figure 13-11. CRC16INIRES Register**

15	14	13	12	11	10	9	8
CRC16INIRES							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
CRC16INIRES							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 13-10. CRC16INIRES Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC16INIRES <sup>(1)</sup>	RW	FFh	CRC16 initialization and result. This register holds the current CRC16 result (according to the CRC16-CCITT standard). Writing to this register initializes the CRC16 calculation with the value written to it.

<sup>(1)</sup> This register is updated with the final signature value one cycle **after** the last data input value is written to the CRC32DI or CRC32DIRB registers. Application should wait for this one cycle delay before reading the result.

### 13.3.0.10 CRC16RESR Register

CRC16 Result Reverse Register

This register always reflects the latest signature of the values collected so far.

**Figure 13-12. CRC16RESR Register**

15	14	13	12	11	10	9	8
CRC16RESR							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
CRC16RESR							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 13-11. CRC16RESR Register Description**

Bit	Field	Type	Reset	Description
15-0	CRC16RESR <sup>(1)</sup>	RW	FFh	CRC16 reverse result. This register holds the current CRC16 result (according to the CRC16-CCITT standard). The order of bits is reversed to the order of bits in the CRC16INIRES register. For example, CRC16RESR[0] = CRC16INIRES[15], CRC32RESR[10] = CRC16INIRES[5] and so on.

<sup>(1)</sup> This register is updated with the final signature value one cycle **after** the last data input value is written to the CRC32DI or CRC32DIRB registers. Application should wait for this one cycle delay before reading the result.

## AES256 Accelerator

The AES256 accelerator module performs Advanced Encryption Standard (AES) encryption or decryption in hardware. It supports key lengths of 128 bits, 192 bits, and 256 bits. This chapter describes the AES256 accelerator.

Topic	Page
14.1 AES Accelerator Introduction.....	550
14.2 AES Accelerator Operation .....	551
14.3 AES256 Registers .....	567

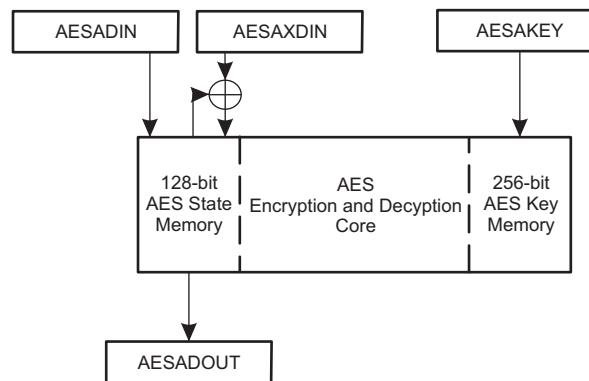
## 14.1 AES Accelerator Introduction

The AES accelerator module performs encryption and decryption of 128-bit data with 128-, 192-, or 256-bit keys according to the advanced encryption standard (AES) (FIPS PUB 197) in hardware.

The AES accelerator features are:

- AES encryption
  - 128 bit in 168 cycles
  - 192 bit in 204 cycles
  - 256 bit in 234 cycles
- AES decryption
  - 128 bit in 168 cycles
  - 192 bit in 206 cycles
  - 256 bit in 234 cycles
- On-the-fly key expansion for encryption and decryption
- Offline key generation for decryption
- Shadow register storing the initial key for all key lengths
- DMA support for ECB, CBC, OFB, and CFB cipher modes
- Byte and half-word access to key, input data, and output data
- AES ready interrupt flag

Figure 14-1 shows the AES accelerator block diagram.



**Figure 14-1. AES Accelerator Block Diagram**

## 14.2 AES Accelerator Operation

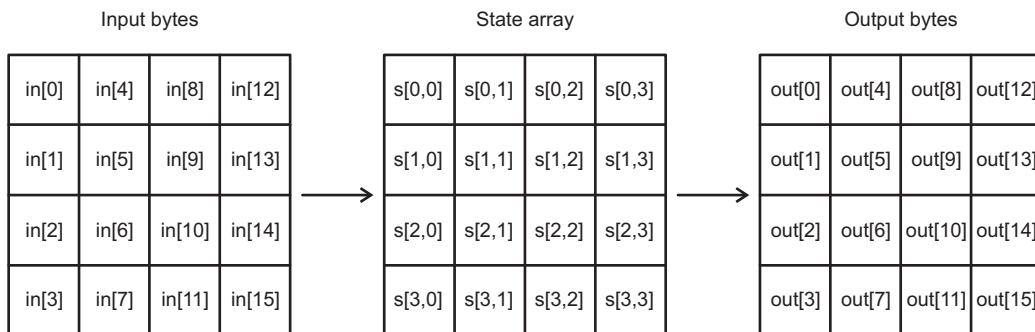
The AES accelerator is configured with user software. The bit AESKLx determines if AES128, AES192, or AES256 is going to be performed. There are four different operation modes available, selectable by the AESOPx bits (see [Table 14-1](#)).

**Table 14-1. AES Operation Modes Overview**

AESOPx	AESKLx	Operation	Clock Cycles
00	00	AES128 encryption	168
	01	AES192 encryption	204
	10	AES256 encryption	234
01	00	AES128 decryption (with initial roundkey) is performed	215
	01	AES192 decryption (with initial roundkey) is performed	255
	10	AES256 decryption (with initial roundkey) is performed	292
10	00	AES128 encryption key schedule is performed	53
	01	AES192 encryption key schedule is performed	57
	10	AES256 encryption key schedule is performed	68
11	00	AES128 (with last roundkey) decryption is performed	168
	01	AES192 (with last roundkey) decryption is performed	206
	10	AES256 (with last roundkey) decryption is performed	234

[Table 14-1](#) lists the execution time of the different modes of operation. While the AES module is operating, the AESBUSY bit is 1. As soon as the operation has finished, the AESRDYIFG bit is set.

Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes called the State. The State consists of four rows of bytes, each containing four bytes, independently if AES128, AES192, or AES256 is performed. [Figure 14-2](#) shows how the input is assigned to the State array, with in[0] being the first data byte written into one of the AES accelerators input registers (AESADIN, AESAXDIN, and AESAXIN). The encrypt or decrypt operations are then conducted on the State array, after which its final values can be read from the output with out[0] being the first data byte read from the AES accelerator data output register (AESADOUT).



**Figure 14-2. AES State Array Input and Output**

If an encryption is to be performed, the initial state is called plaintext. If a decryption is to be performed, the initial state is called ciphertext.

The module allows half-word and byte access to all data registers—AESAKEY, AESADIN, AESAXDIN, AESAXIN, and AESADOUT. Half-word and byte access cannot be mixed while reading from or writing into one of the registers. However, it is possible to write one of the registers using byte access and another using half-word access.

**NOTE: General Access Restrictions**

While the AES accelerator is busy (AESBUSY = 1):

- AESADOUT always reads as zero.
- The AESDOUTCNTx counter, the AESDOUTRD flag, and the AESDINWR flag are reset.
- Any attempt to change AESOPx, AESKLx, AESDINWR, or AESKEYWR is ignored.
- Writing to AESAKEY, AESADIN, AESAXDIN, or AESAXIN aborts the current operation, the complete module is reset (except for the AESRDYIE, AESOPx, and AESKLx), and the AES error flag AESERRFG is set.

AESADIN, AESAXDIN, AESAXIN, and AESAKEY are write-only registers and always read as zero.

Writing data into AESADIN, AESAXDIN, or AESAXIN influences the content of the corresponding output data; for example, writing in[0] alters out[0], writing in[1] alters out[1], and so on. However, interleaved operation is possible; for example, first reading out[0] and then writing in[0], and continuing with reading out[1] and then writing in[1], and so on. This interleaved operation must be either byte or half-word access on in[x] and out[x].

**Access Restriction With Cipher Modes Enabled (AEPCMEN = 1)**

When cipher modes are enabled (AEPCMEN = 1) and a cipher block operation is being processed (AESBLKCNTx > 0), writes to the following bits are ignored, independent of AESBUSY: AEPCMEN, AEPCMx, AESKLx, AESOPx, and AESBLKCNTx. Writing to AESAKEY aborts the cipher block mode operation if AESBUSY = 1, and the complete module is reset (except for AESRDYIE, AESOPx, and AESKLx).

#### 14.2.1 Load the Key (128-Bit, 192-Bit, or 256-Bit Keylength)

The key can be loaded by writing to the AESAKEY register or by setting AESKEYWR. Depending on the selected keylength (AESKLx), a different number of bits must be loaded:

- If AESKLx = 00, the 128-bit key must be loaded using either 16 byte-writes or 8 half-word-writes to AESAKEY.
- If AESKLx = 01, the 192-bit key must be loaded using either 24 byte-writes or 12 half-word-writes to AESAKEY.
- If AESKLx = 10, the 256-bit key must be loaded using either 32 byte-writes or 16 half-word-writes to AESAKEY.

The key memory is reset after changing the AESKLx.

If a key was loaded previously without changing AESOPx, the AESKEYWR flag is cleared with the first write access to AESAKEY.

If the conversion is triggered without writing a new key, the last key is used. The key must always be written before writing the data.

#### 14.2.2 Load the Data (128-Bit State)

The state can be loaded by writing to AESADIN, AESAXDIN, or AESAXIN with 16 byte writes or 8 half-word writes. Do not mix byte and half-word mode when writing the state. Writing to a mixture of AESADIN, AESAXDIN, and AESAXIN using the same byte or half-word data format is allowed. When the 16th byte or 8th half-word of the state is written, AESDINWR is set.

When writing to AESADIN, the corresponding byte or half-word of the state is overwritten. If AESADIN is used to write the last byte or half-word of the state, encryption or decryption starts automatically.

When writing to AESAXDIN, the corresponding byte or half-word is XORed with the current byte or half-word of the state. If AESAXDIN is used to write the last byte or half-word of the state, encryption or decryption starts automatically.

Writing to AESAXIN has the same behavior as writing to AESAXDIN: the corresponding byte or half-word is XORed with the current byte or half-word of the state; however, writing the last byte or half-word of the state using AESAXIN does not start encryption or decryption.

#### 14.2.3 *Read the Data (128-Bit State)*

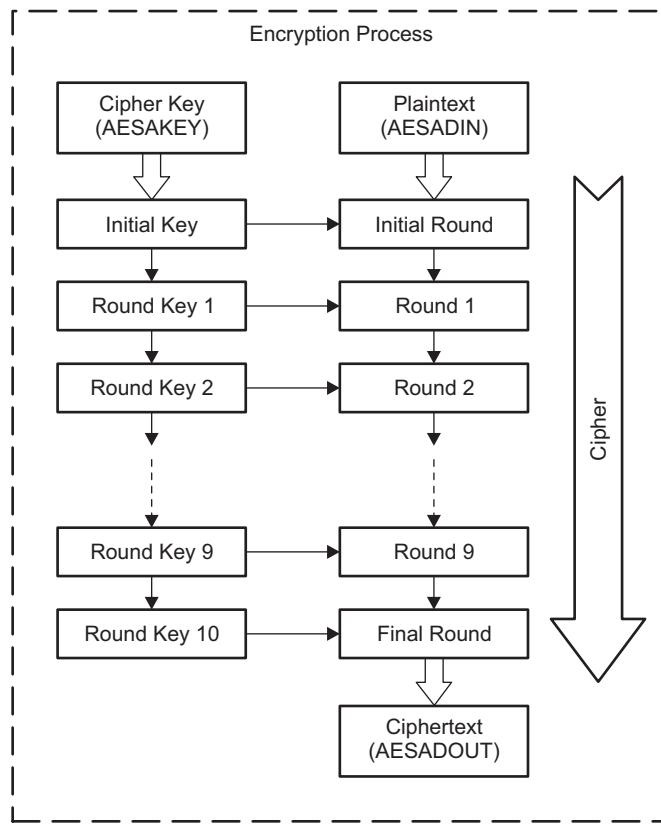
The state can be read if AESBUSY = 0 using 16 byte reads or 8 half-word reads to AESADOUT. When all 16 bytes are read, the AESDOUTRD flag indicates completion.

#### 14.2.4 *Trigger an Encryption or Decryption*

The AES module's encrypt or decrypt operations are triggered if the state was completely written to the AESADIN or AESAXDIN registers. Alternatively, the bit AESDINWR can be set to trigger a operation if AESCMEN = 0.

### **14.2.5 Encryption**

Figure 14-3 shows the encryption process with the cipher being a series of transformations that convert the plaintext written into the AESADIN register to a ciphertext that can be read from the AESADOUT register using the cipher key provided in the AESAKEY register.



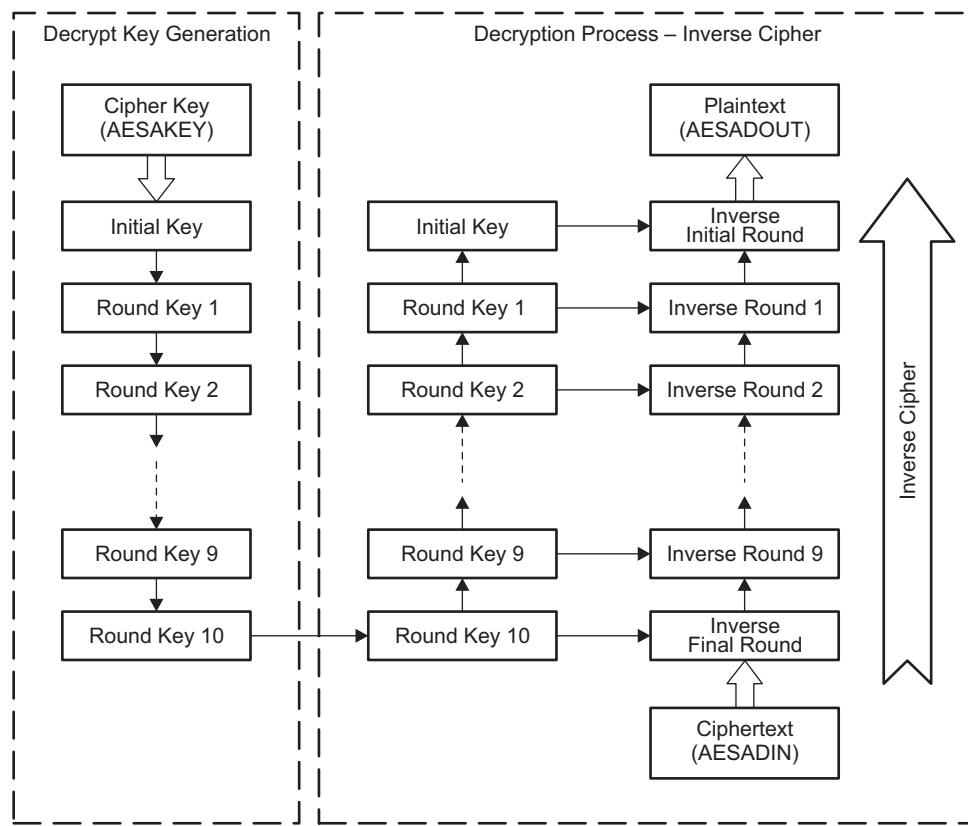
**Figure 14-3. AES Encryption Process for 128-Bit Key**

To perform encryption:

1. Set AESOPx = 00 to select encryption. Changing the AESOPx bits clears the AESKEYWR flag, and a new key must be loaded in the next step.
  2. Load the key as described in [Section 14.2.1](#).
  3. Load the state (data) as described in [Section 14.2.2](#). After the data is loaded, the AES module starts the encryption.
  4. After the encryption is ready, the result can be read from AESADOUT as described in [Section 14.2.3](#).
  5. To encrypt additional data with the same key loaded in step 2, write the new data into AESADIN after the results of the operation on the previous data were read from AESADOUT. When an additional 16 data bytes are written, the module automatically starts the encryption using the key loaded in step 2. When implementing, for example, the output feedback (OFB) cipher block chaining mode, setting the AESDINWR flag triggers the next encryption, and the module starts the encryption using the output data from the previous encryption as the input data.

### **14.2.6 Decryption**

Figure 14-4 shows the decryption process with the inverse cipher being a series of transformations that convert the ciphertext written into the AESADIN register to a plaintext that can be read from the AESADOUT register using the cipher key provided in the AESAKEY register.



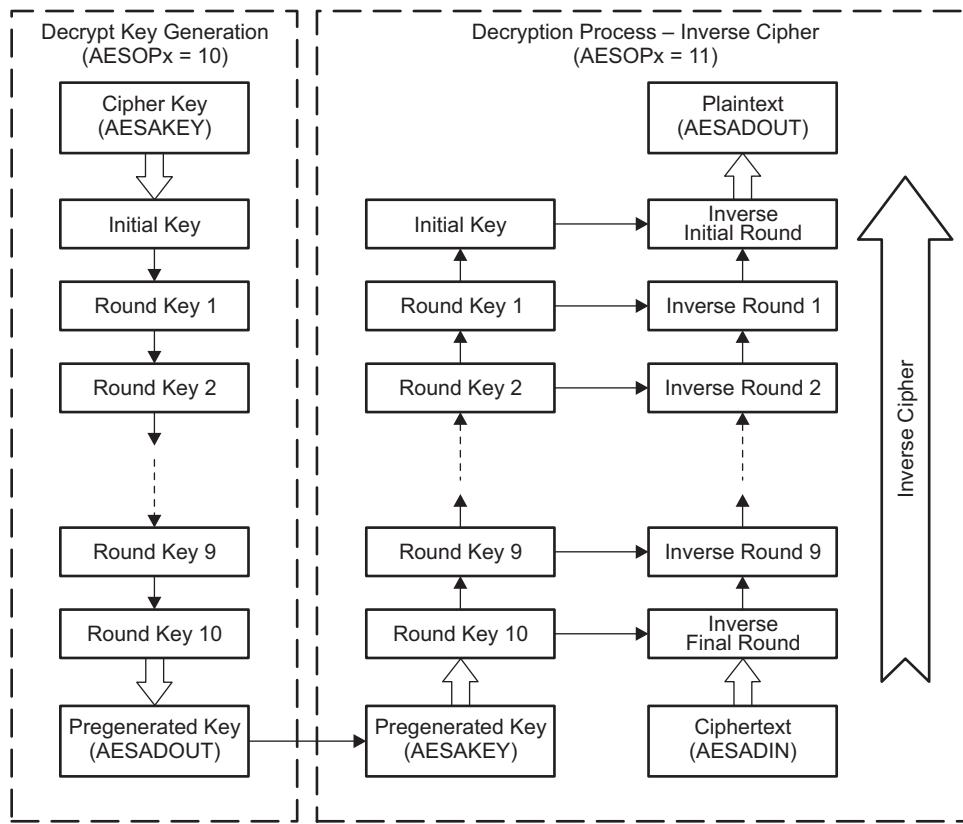
**Figure 14-4. AES Decryption Process Using AESOPx = 01 for 128-Bit Key**

The steps to perform decryption are:

1. Set AESOPx = 01 to select decryption using the same key used for encryption. Set AESOPx = 11 if the first-round key required for decryption (the last roundkey) is already generated and will be loaded in step 2. Changing the AESOPx bits clears the AESKEYWR flag, and a new key must be loaded in step 2.
2. Load the key according to [Section 14.2.1](#).
3. Load the state (data) according to [Section 14.2.2](#). After the data is loaded, the AES module starts the decryption.
4. After the decryption is ready, the result can be read from AESADOUT according to [Section 14.2.3](#).
5. If additional data should be decrypted with the same key loaded in step 2, new data can be written into AESADIN after the results of the operation on the previous data were read from AESADOUT. When additional 16 data bytes are written, the module automatically starts the decryption using the key loaded in step 2.

#### 14.2.7 Decryption Key Generation

[Figure 14-5](#) shows the decryption process with a pregenerated decryption key. In this case, the decryption key is calculated first with AESOPx = 10, then the precalculated key can be used together with the decryption operation AESOPx = 11.



**Figure 14-5. AES Decryption Process Using AESOPx = 10 and 11 for 128-Bit Key**

To generate the decryption key independent from the actual decryption:

1. Set AESOPx = 10 to select decryption key generation. Changing the AESOPx bits clears the AESKEYWR flag, and a new key must be loaded in step 2.
  2. Load the key as described in [Section 14.2.1](#). The generation of the first round key required for decryption is started immediately.
  3. While the AES module is performing the key generation, the AESBUSY bit is 1. 53 CPU clock cycles are required to complete the key generation for a 128-bit key (for other key lengths, see [Table 14-1](#)). After its completion, the AESRDYIFG is set, and the result can be read from AESADOUT. When all 16 bytes are read, the AESDOUTRD flag indicates completion.
- The AESRDYIFG flag is cleared when reading AESADOUT or writing to AESAKEY or AESADIN.
4. If data should be decrypted with the generated key, AESOPx must be set to 11. Then the generated key must be loaded or, if it was just generated with AESOPx = 10, set the AESKEYWR flag by software to indicate that the key is already valid.
  5. See [Section 14.2.6](#) for instructions on the decryption steps.

#### 14.2.8 AES Key Buffer

The AES128, AES192, or AES256 algorithm operates not only on the state, but also on the key. To avoid the need of reloading the key for each encryption or decryption, a key buffer is included in the AES accelerator.

### 14.2.9 Using the AES Accelerator With Low-Power Modes

The AES accelerator module provides automatic clock activation for MCLK for use with low-power modes. When the AES accelerator is busy, it automatically activates MCLK, regardless of the control-bit settings for the clock source. The clock remains active until the AES accelerator completes its operation.

The interrupt flag AESRDYIFG is reset after a Hard Reset or with AESSWRST = 1. AESRDYIE is reset after a Hard Reset but is not reset by AESSWRST = 1.

### 14.2.10 AES Accelerator Interrupts

The AESRDYIFG interrupt flag is set when the AES module completes the selected operation on the provided data. An interrupt request is generated if AESRDYIE is set. AESRDYIFG is automatically reset if the AES interrupt is serviced, if AESADOUT is read, or if AESADIN or AESAKEY are written. AESRDYIFG is reset after a Hard Reset or with AESSWRST = 1. AESRDYIE is reset after a Hard Reset but is not reset by AESSWRST = 1.

### 14.2.11 DMA Operation and Implementing Block Cipher Modes

DMA operation, meaning the implementation of the ciphers Electronic code book (ECB), Cipher block chaining (CBC), Output feedback (OFB), and Cipher feedback (CFB) using the DMA, supports easy and fast encryption and decryption of more than 128 bits.

When DMA ciphermode support is enabled by setting the AESCMEN bit, the AES256 module triggers 'AES trigger 0', 'AES trigger 1', and 'AES trigger 2' (also called 'AES trigger 0-2') in a certain order to execute different block cipher modes together with the DMA module.

For example, when using ECB encryption with AESCMEN = 1, 'AES trigger 0' is triggered eight times for DMA half-word access to read out AESADOUT, and then 'AES trigger 1' is triggered eight times to fill the next data into AESADIN.

Table 14-2 shows the behavior of the 'AES trigger 0-2' for the different ciphermodes selected by AESCMx.

**Table 14-2. 'AES trigger 0-2' Operation When AESCMEN = 1**

AESCMx	AESOPx	'AES trigger 0'	'AES trigger 1'	'AES trigger 2'
00 ECB	00 encryption	Set after encryption ready, set again until 128 bit are read from AESADOUT	Set to load the first block and set after 'AES trigger 0' was served the last time, set again until 128 bit are written to AESADIN	not set
	01 or 11 decryption	Set after decryption ready, set again until 128 bit are read from AESADOUT	Set to load the first block and set after 'AES trigger 0' was served the last time, set again until 128 bit are written to AESADIN	not set
01 CBC	00 encryption	Set after encryption ready, set again until 128 bit are read from AESADOUT	Set after 'AES trigger 0' was served the last time, set again until 128 bit are written to AESADIN	not set
	01 or 11 decryption	Set after decryption ready, set again until 128 bit are written to from AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit read from AESADOUT	Set after 'AES trigger 1' was served the last time, set again until 128 bit are written to AESADIN
10 OFB	00 encryption	Set after encryption ready, set again until 128 bit are written to AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit are read from AESADOUT	Set after 'AES trigger 1' was served the last time, set again until 128 bit are written to AESADIN
	01 or 11 decryption	Set after decryption ready, set again until 128 bit are written to AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit are read from AESADOUT	Set after 'AES trigger 1' was served the last time, set again until 128 bit are written to AESADIN

**Table 14-2. 'AES trigger 0-2' Operation When AESCMEN = 1 (continued)**

AESCMx	AESOPx	'AES trigger 0'	'AES trigger 1'	'AES trigger 2'
11 CFB	00 encryption	Set after encryption ready, set again until 128 bit are written to AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit are read from AESADOUT	not set
	01 or 11 decryption	Set after decryption ready, set again until 128 bit are written to AESAXIN	Set after 'AES trigger 0' was served the last time, set again until 128 bit are read from AESADOUT	Set after 'AES trigger 1' was served the last time, set again until 128 bit are written to AESADIN

The retriggering of the 'AES trigger 0-2' until 128-bit of data are written or read from the corresponding register supports both byte and half-word access for writing and reading the state through the DMA.

For AESCMEN = 0, no DMA triggers are generated.

The following sections explains the configuration of the AES module for automatic cipher mode execution using DMA.

It is assumed that the key is written by software (or by a separate DMA transfer) before writing the first block to the AES state. The key shadow register always restores the original key, so that there is no need to reload it. The AESAKEY register should not be written after AESBLKCNTx is written to a non-zero value.

The number of blocks to be encrypted or decrypted must be programmed into the AESBLKCNTx bits prior to writing the first data. Writing a non-zero value into AESBLKCNTx starts the cipher mode sequence and, thus, AESBLKCNTx must be written after the DMA channels are configured.

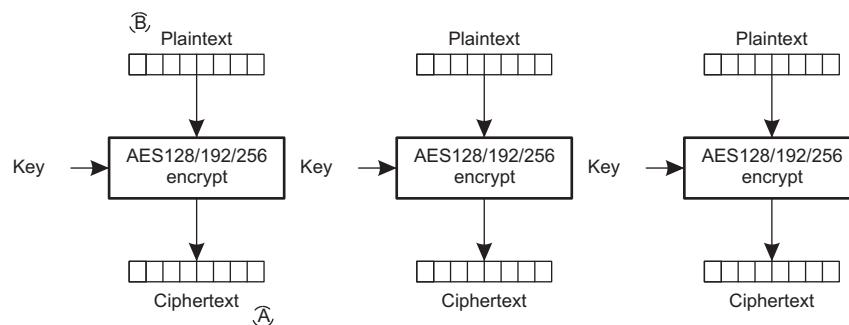
Throughout these sections, the different DMA channels are called DMA\_A, DMA\_B, and so on. In the figures, these letters appear in dotted circles showing which operation is going to be executed by which DMA channel. The DMA counter must be loaded with a multiple of 8 for half-word mode or a multiple of 16 for byte mode. The DMA priorities of DMA\_A, DMA\_B, and DMA\_C do not play any role.

#### 14.2.11.1 Electronic Codebook (ECB) Mode

The electronic codebook block cipher mode is the most simple cipher mode. The data is divided into 128-bit blocks and encrypted and decrypted separately.

The disadvantage of the ECB is that the same 128-bit plaintext is always encrypted to the same ciphertext, whereas the other modes encrypt each block differently, partly dependent on the already executed encryptions.

##### 14.2.11.1.1 ECB Encryption



**Figure 14-6. ECB Encryption**

To implement the ECB encryption without CPU interaction, two DMA channels are needed. Static DMA priorities must be enabled.

**Table 14-3. AES and DMA Configuration for ECB Encryption**

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'
1	00	00	Read ciphertext from AESADOUT	Write plaintext to AESADIN, which also triggers the next encryption

The following pseudo code snippet shows the implementation of the ECB encryption in software:

```

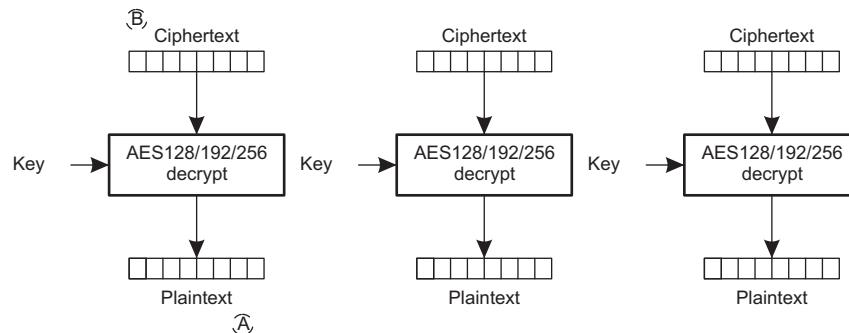
ECB_Encryption(key, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Configure AES for block cipher:
    AESCMEN= 1; AESCMx= ECB; AESOPx= 00;

    Write key into AESAKEY;
    Setup DMA:
        DMA0: Triggered by AES trigger 0,
              Source: AESADOUT, Destination: ciphertext, Size: num_blocks*8 half-words
        DMA1: Triggered by AES trigger 1,
              Source: plaintext, Destination: AESADIN,      Size: num_blocks*8 half-words

    Start encryption:
    AESBLKCNT= num_blocks;

    End of encryption: DMA0IFG=1
}
  
```

#### 14.2.11.1.2 ECB Decryption



**Figure 14-7. ECB Decryption**

**Table 14-4. AES DMA Configuration for ECB Decryption**

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'
1	00	01 or 11	Read plaintext from AESADOUT	Write ciphertext to AESADIN, which also triggers the next decryption

The following pseudo code snippet shows the implementation of the ECB decryption in software:

```

ECB_Decryption(key, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Generate Decrypt Key
    Configure AES:
        AESCMEN= 0; AESOPx= 10;
        Write key into AESKEY;
        Wait until key generation completed.

    Configure AES for block cipher:
        AESCMEN= 1; AESCMx= ECB; AESOPx= 11;
        AESKEYWR= 1; // Use previously generated key

    Setup DMA:
        DMA0: Triggered by AES trigger 0,
            Source: AESADOUT, Destination: plaintext, Size: num_blocks*8 half-words
        DMA1: Triggered by AES trigger 1,
            Source: ciphertext, Destination: AESADIN, Size: num_blocks*8 half-words

    Start decryption:
        AESBLKCNT= num_blocks;

    End of decryption: DMA0IFG=1
}

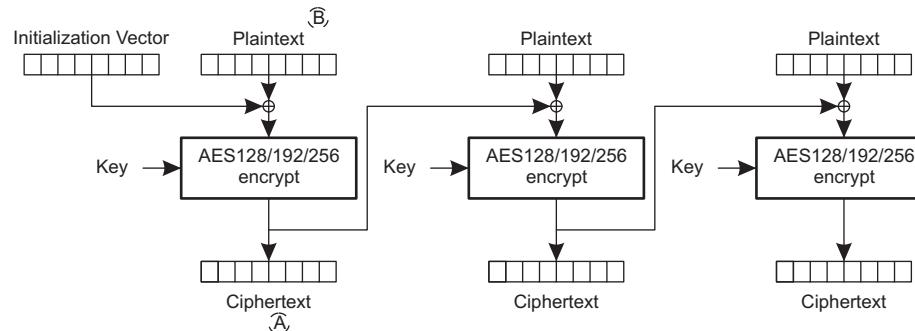
```

### 14.2.11.2 Cipher Block Chaining (CBC) Mode

The cipher block chaining ciphermode always performs an XOR on the ciphertext of the previous block with the current block. Therefore, the encryption of each block depends on not only the key but also on the previous encryption.

#### 14.2.11.2.1 CBC Encryption

For encryption, the initialization vector must be loaded by software (or by a separate DMA transfer) into AESXIN before the DMA can be enabled to write the first 16 bytes of the plaintext into AESAXDIN (see Figure 14-8).



**Figure 14-8. CBC Encryption**

**Table 14-5. AES and DMA Configuration for CBC Encryption**

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'
1	01	00	Read ciphertext from AESADOUT	Write plaintext to AESAXDIN, which also triggers the next encryption

The following pseudo code snippet shows the implementation of the CBC encryption in software:

```

CBC_Encryption(key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
    AESWRST= 1;
    Configure AES for block cipher:
    AESCMEN= 1; AESCMx= CBC; AESOPx= 00;

    Write key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                           // Assumes that state is reset (=> XORing with Zeros).
    Setup DMA:
    DMA0: Triggered by AES trigger 0,
           Source: AESADOUT, Destination: ciphertext, Size: num_blocks*8 half-words
    DMA1: Triggered by AES trigger 1,
           Source: plaintext, Destination: AESAXDIN,   Size: num_blocks*8 half-words

    Start encryption:
    AESBLKCNT= num_blocks;

    End of encryption: DMA0IFG=1
}
  
```

#### 14.2.11.2.2 CBC Decryption

For CBC decryption, the first block of data needs special handling because the output must be XORed with the Initialization Vector. For that purpose, the DMA triggered by 'AES trigger 0' must be configured to read the data from the Initialization Vector first and then must be reconfigured to read from the ciphertext.

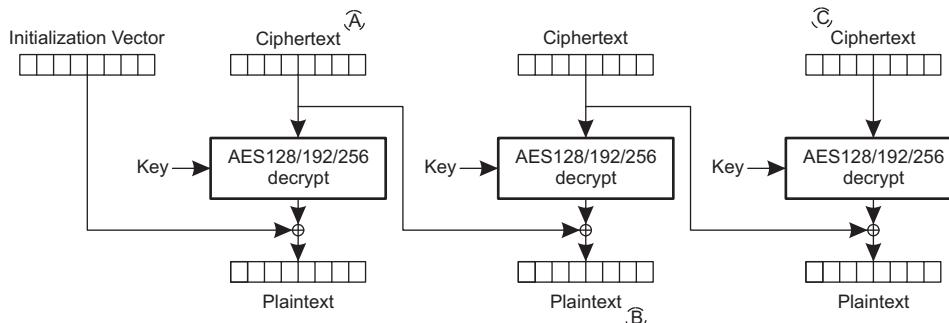


Figure 14-9. CBC Decryption

Table 14-6. AES and DMA Configuration for CBC Decryption

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'	DMA_C Triggered by 'AES trigger 2'
1	01	01 or 11	Write the previous ciphertext block to AESAXIN	Read plaintext from AESADOUT	Write next plaintext to AESADIN, which also triggers the next decryption

The following pseudo code snippet shows the implementation of the CBC decryption in software:

```
CBC_Decryption(key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Generate Decrypt Key:
    Configure AES:
        AESCMEN= 0; AESOPx= 10;
        Write key into AESKEY;
        Wait until key generation completed;

    Configure AES for block cipher:
        AESCMEN= 1; AESCMx= CBC; AESOPx= 11;
        AESKEYWR= 1; // Use previously generated key

    Setup DMA:
        DMA0: Triggered by AES trigger 0,
              Source: IV, Destination: AESAXIN, Size: 8 half-words
        DMA1: Triggered by AES trigger 1,
              Source: AESADOUT, Destination: plaintext, Size: num_blocks*8 half-words
        DMA2: Triggered by AES trigger 2,
              Source: ciphertext, Destination: AESADIN, Size: num_blocks*8 half-words

    Start decryption:
        AESBLKCNT= num_blocks;

    Wait until first block is decrypted: DMA0IFG=1;

    Setup DMA0 for further blocks:
        DMA0: // Write previous cipher text into AES module
              Triggered by AES trigger 0,
              Source: ciphertext, Destination: AESAXIN, Size: (num_blocks-1)*8 half-words

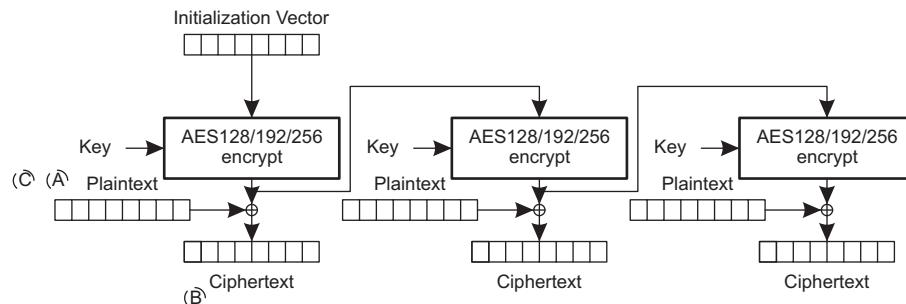
    End of decryption: DMA1IFG=1
}
```

### 14.2.11.3 Output Feedback (OFB) Mode

The output feedback ciphermode continuously encrypts the initialization vector. The ciphertext is generated by XORing the corresponding plaintext with the encrypted initialization vector.

The initialization vector must be loaded by software (or by a separate DMA transfer).

#### 14.2.11.3.1 OFB Encryption



**Figure 14-10. OFB Encryption**

**Table 14-7. AES and DMA Configuration for OFB Encryption**

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'	DMA_C Triggered by 'AES trigger 2'
1	10	00	Write the plaintext of the current block to AESAXIN	Read ciphertext from AESADOUT	Write the plaintext of the current block to AESAXDIN, which also triggers the next encryption

The following pseudo code snippet shows the implementation of the OFB encryption in software:

```

OFB_Encryption(Key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
    AECSRST= 1;
    Configure AES:
    AESCMEN= 1; AESCMx= OFB; AESOPx= 00;

    Write Key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                           // Assumes that state is reset (=> XORing with Zeros).

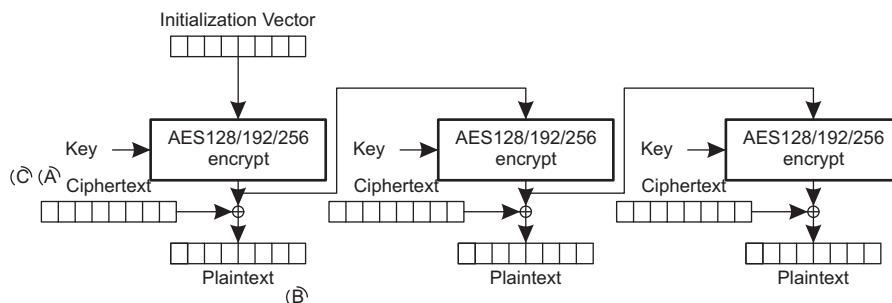
    Setup DMA:
    DMA0: Triggered by AES trigger 0,
          Source: plaintext, Destination: AESAXIN,      Size: num_blocks*8 half-words
    DMA1: Triggered by AES trigger 1,
          Source: AESADOUT, Destination: ciphertext, Size: num_blocks*8 half-words
    DMA2: Triggered by AES trigger 2,
          Source: plaintext, Destination: AESAXDIN,     Size: num_blocks*8 half-words

    Start encryption:
    AESBLKCNT= num_blocks;
    Trigger encryption by setting AESDINWR= 1;

    End of encryption: DMA1IFG=1
}

```

#### 14.2.11.3.2 OFB Decryption



**Figure 14-11. OFB Decryption**

**Table 14-8. AES and DMA Configuration for OFB Decryption**

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'	DMA_C Triggered by 'AES trigger 2'
1	10	01 or 11 <sup>(1)</sup>	Write the ciphertext of the current block to AESAXIN	Read plaintext from AESADOUT	Write the ciphertext of the current block to AESAXDIN, which also triggers the next encryption

<sup>(1)</sup> In this cipher mode, the decryption also uses AES encryption on block level, thus the key used for decryption is identical with the key used for encryption; therefore, no decryption key generation is required.

The following pseudo code snippet shows the implementation of the OFB decryption in software:

```

OFB_Decryption(Key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
    AECSRST= 1;
    Configure AES:
    AESCMEN= 1; AESCMx= OFB; AESOPx= 01;

    Write Key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                           // Assumes that state is reset (=> XORing with Zeros).
    Setup DMA:
    DMA0: Triggered by AES trigger 0,
          Source: ciphertext, Destination: AESAXIN,   Size: num_blocks*8 half-words
    DMA1: Triggered by AES trigger 1,
          Source: AESADOUT,   Destination: plaintext, Size: num_blocks*8 half-words
    DMA2: Triggered by AES trigger 2,
          Source: ciphertext, Destination: AESAXDIN,  Size: num_blocks*8 half-words

    Start decryption:
    AESBLKCN= num_blocks;
    Trigger decryption by setting AESDINWR= 1;

    End of decryption: DMA1IFG=1
}

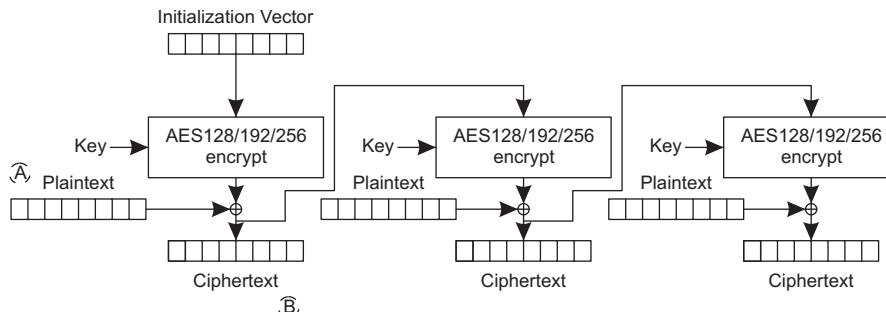
```

#### 14.2.11.4 Cipher Feedback (CFB) Mode

In the cipher feedback ciphermode, the plaintext of the new block is XORed to the last encryption result. The result of the encryption is the input for the new encryption.

The initialization vector must be loaded by software (or by a separate DMA transfer).

##### 14.2.11.4.1 CFB Encryption



**Figure 14-12. CFB Encryption**

**Table 14-9. AES and DMA Configuration for CFB Encryption**

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'
1	11	00	Write the plaintext of the current block to AESAXIN	Read the ciphertext from AESADOUT, which also triggers the next encryption

The following pseudo code snippet shows the implementation of the CFB encryption in software:

```

CFB_Encryption(Key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
    AESWRST= 1;
    Configure AES:
    AESCMEN= 1; AESCMx= CFB; AESOPx= 00;

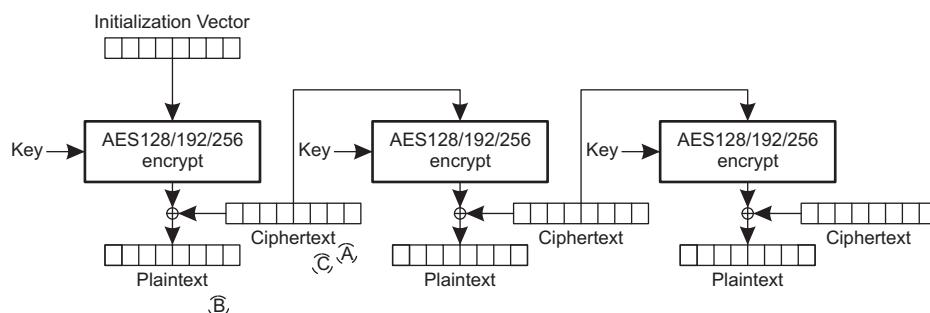
    Write Key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                           // Assumes that state is reset (=> XORing with Zeros).
    Setup DMA:
    DMA0: Triggered by AES trigger 0,
          Source: plaintext, Destination: AESAXIN,      Size: num_blocks*8 half-words
    DMA1: Triggered by AES trigger 1,
          Source: AESADOUT,   Destination: ciphertext, Size: num_blocks*8 half-words

    Start encryption:
    AESBLKCNT= num_blocks;
    Trigger encryption by setting AESDINWR= 1;

    End of encryption: DMA1IFG=1
}

```

#### 14.2.11.4.2 CFB Decryption



**Figure 14-13. CFB Decryption**

**Table 14-10. AES and DMA Configuration for CFB Decryption**

AES CMEN	AES CMx	AES OPx	DMA_A Triggered by 'AES trigger 0'	DMA_B Triggered by 'AES trigger 1'	DMA_C Triggered by 'AES trigger 2'
1	11	01 or 11 <sup>(1)</sup>	Write the ciphertext of the current block to AESAXIN	Read the plaintext from AESADOUT	Write the ciphertext of the current block to AESADIN, which also triggers the next encryption

<sup>(1)</sup> In this cipher mode, the decryption also uses AES encryption on block level thus the key used for decryption is identical with the key used for encryption; therefore, no decryption key generation is required.

The following pseudo code snippets shows the implementation of the CFB encryption and decryption in software:

```

CFB_Decryption(Key, IV, plaintext, ciphertext, num_blocks)
// Pseudo Code
{
    Reset AES Module (clears internal state memory):
    AESWRST= 1;
    Configure AES:
    AESCMEN= 1; AESCMx= CFB; AESOPx= 01;

    Write Key into AESAKEY;
    Write IV into AESAXIN; // Does not trigger encryption.
                           // Assumes that state is reset (=> XORing with Zeros).
    Setup DMA:
    DMA0: Triggered by AES trigger 0,
           Source: ciphertext, Destination: AESAXIN, Size: num_blocks*8 half-words
    DMA1: Triggered by AES trigger 1,
           Source: AESADOUT, Destination: plaintext, Size: num_blocks*8 half-words
    DMA2: Triggered by AES trigger 2,
           Source: ciphertext, Destination: AESADIN, Size: num_blocks*8 half-words

    Start decryption:
    AESBLKCNT= num_blocks;
    Trigger decryption by setting AESDINWR= 1;

    End of decryption: DMA1IFG=1
}

```

## 14.3 AES256 Registers

**Table 14-11** lists the memory-mapped registers for the AES256 module. See the device-specific data sheet for the base address of the module. All offset addresses not listed in this table should be considered as reserved locations and the register contents should not be modified.

**Table 14-11. AES256 Registers**

Offset	Acronym	Register Name	Section
00h	AESACTL0	AES accelerator control register 0	<a href="#">Section 14.3.1</a>
02h	AESACTL1	AES accelerator control register 1	<a href="#">Section 14.3.2</a>
04h	AESASTAT	AES accelerator status register	<a href="#">Section 14.3.3</a>
06h	AESAKEY	AES accelerator key register	<a href="#">Section 14.3.4</a>
08h	AESADIN	AES accelerator data in register	<a href="#">Section 14.3.5</a>
0Ah	AESADOUT	AES accelerator data out register	<a href="#">Section 14.3.6</a>
0Ch	AESAXDIN	AES accelerator XORed data in register	<a href="#">Section 14.3.7</a>
0Eh	AESAXIN	AES accelerator XORed data in register (no trigger)	<a href="#">Section 14.3.8</a>

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 14.3.1 AESACTL0 Register

AES Accelerator Control Register 0

**Figure 14-14. AESACTL0 Register**

15	14	13	12	11	10	9	8
AESCMEN	Reserved		AESRDYIE	AESERRFG	Reserved		AESRDYIFG
rw-0	r-0	r-0	rw-0	rw-0	r-0	r-0	rw-0
7	6	5	4	3	2	1	0
AESSWRST	AESCMx		Reserved	AESKLx		AESOPx	
rw-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0
			Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0.				

**Table 14-12. AESACTL0 Register Description**

Bit	Field	Type	Reset	Description
15	AESCMEN	RW	0h	AESCMEN enables the support of the ciphermodes ECB, CBC, OFB and CFB together with the DMA. Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0. 0 = No DMA triggers are generated 1 = DMA ciphermode support operation is enabled and the corresponding DMA triggers are generated.
14-13	Reserved	R	0h	Reserved
12	AESRDYIE	RW	0h	AES ready interrupt enable. AESRDYIE is not reset by AESSWRST = 1. 0b = Interrupt disabled 1b = Interrupt enabled
11	AESERRFG	RW	0h	AES error flag. AESAKEY or AESADIN were written while an AES operation was in progress. The bit must be cleared by software. 0b = No error 1b = Error occurred
10-9	Reserved	R	0h	Reserved
8	AESRDYIFG	RW	0h	AES ready interrupt flag. Set when the selected AES operation was completed and the result can be read from AESADOUT. Automatically cleared when AESADOUT is read or AESAKEY or AESADIN is written. 0b = No interrupt pending 1b = Interrupt pending
7	AESSWRST	RW	0h	AES software reset. Immediately resets the complete AES accelerator module even when busy except for the AESRDYIE, the AESKLx and the AESOPx bits. It also clears the (internal) state memory. The AESSWRST bit is automatically reset and is always read as zero. 0b = No reset 1b = Reset AES accelerator module
6-5	AESCMx	RW	0h	AES cipher mode select. These bits are ignored for AESCMEN = 0. Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0. 00b = ECB 01b = CBC 10b = OFB 11b = CFB
4	Reserved	R	0h	Reserved
3-2	AESKLx	RW	0h	AES key length. These bits define which of the 3 AES standards is performed. The AESKLx bits are not reset by AESSWRST = 1. Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0. 00b = AES128. The key size is 128 bit. 01b = AES192. The key size is 192 bit. 10b = AES256. The key size is 256 bit. 11b = Reserved

**Table 14-12. AESACTL0 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
1-0	AESOPx	RW	0h	<p>AES operation. The AESOPx bits are not reset by AECSRST = 1. Writes are ignored when AESCMEN = 1 and AESBLKCNTx &gt; 0.</p> <p>00b = Encryption</p> <p>01b = Decryption. The provided key is the same key used for encryption.</p> <p>10b = Generate first round key required for decryption.</p> <p>11b = Decryption. The provided key is the first round key required for decryption.</p>

### 14.3.2 AESACTL1 Register

AES Accelerator Control Register 1

**Figure 14-15. AESACTL1 Register**

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
AESBLKCNTx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0.							

**Table 14-13. AESACTL1 Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved. Always reads 0.
7-0	AESBLKCNTx	RW	0h	Cipher Block Counter. Number of blocks to be encrypted or decrypted with block cipher modes enabled (AESCMEN = 1). Ignored if AESCMEN = 0. The block counter decrements with each performed encryption or decryption. Writes are ignored when AESCMEN = 1 and AESBLKCNTx > 0.

### 14.3.3 AESASTAT Register

AES Accelerator Status Register

**Figure 14-16. AESASTAT Register**

15	14	13	12	11	10	9	8
AESDOUTCNTx				AESDINCNTx			
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
AESKEYCNTx				AESDOUTRD	AESDINWR	AEKEYWR	AESBUSY
r-0	r-0	r-0	r-0	r-0	rw-0	rw-0	r-0

**Table 14-14. AESASTAT Register Description**

Bit	Field	Type	Reset	Description
15-12	AESDOUTCNTx	R	0h	Bytes read through AESADOUT. Reset when AESDOUTRD is reset. If AESDOUTCNTx = 0 and AESDOUTRD = 0, no bytes were read. If AESDOUTCNTx = 0 and AESDOUTRD = 1, all bytes were read.
11-8	AESDINCNTx	R	0h	Bytes written through AESADIN, AESAXDIN or AESAXIN. Reset when AESDINWR is reset. If AESDINCNTx = 0 and AESDINWR = 0, no bytes were written. If AESDINCNTx = 0 and AESDINWR = 1, all bytes were written.
7-4	AESKEYCNTx	R	0h	Bytes written through AESAKEY for AESKLx=00, half-words written through AESAKEY if AESKLx = 01, 10, 11. Reset when AEKEYWR is reset. If AESKEYCNTx = 0 and AEKEYWR = 0, no bytes were written. If AESKEYCNTx = 0 and AEKEYWR = 1, all bytes were written.
3	AESDOUTRD	R	0h	All 16 bytes read from AESADOUT. AESDOUTRD is reset by PUC, AECSRST, an error condition, changing AESOPx, changing AESKLx, when the AES accelerator is busy, and when the output data is read again. 0 = Not all bytes read 1 = All bytes read
2	AESDINWR	RW	0h	All 16 bytes written to AESADIN, AESAXDIN or AESAXIN. This bit can be modified by software only if AESCMEN=0. Changing its state by software also resets the AESDINCNTx bits. AESDINWR is reset by PUC, AECSRST, an error condition, changing AESOPx, changing AESKLx, the start to (over)write the data, and when the AES accelerator is busy. Because it is reset when AESOPx or AESKLx is changed it can be set by software again to indicate that the current data is still valid. 0 = Not all bytes written 1 = All bytes written
1	AEKEYWR	RW	0h	All 16 bytes written to AESAKEY. This bit can be modified by software but it must not be reset by software (1→0) if AESCMEN=1. Changing its state by software also resets the AESKEYCNTx bits. AEKEYWR is reset by PUC, AECSRST, an error condition, changing AESOPx, changing AESKLx, and the start to (over)write a new key. Because it is reset when AESOPx is changed it can be set by software again to indicate that the loaded key is still valid 0 = Not all bytes written 1 = All bytes written
0	AESBUSY	R	0h	AES accelerator module busy; encryption, decryption, or key generation in progress. 0 = Not busy 1 = Busy

#### 14.3.4 AESAKEY Register

AES Accelerator Key Register

**Figure 14-17. AESAKEY Register**

15	14	13	12	11	10	9	8
AESKEY1x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0
7	6	5	4	3	2	1	0
AESKEY0x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0

**Table 14-15. AESAKEY Register Description**

Bit	Field	Type	Reset	Description
15-8	AESKEY1x	W	0h	AES key byte n+1 when AESAKEY is written as half-word. Do not use these bits for byte access. Do not mix half-word and byte access. Always reads as zero. The key is reset by PUC or by AECSRST = 1.
7-0	AESKEY0x	W	0h	AES key byte n when AESAKEY is written as half-word. AES next key byte when AESAKEY_L is written as byte. Do not mix half-word and byte access. Always reads as zero. The key is reset by PUC or by AECSRST = 1.

### **14.3.5 AESADIN Register**

AES Accelerator Data In Register

**Figure 14-18. AESADIN Register**

15	14	13	12	11	10	9	8
AESDIN1x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0
7	6	5	4	3	2	1	0
AESDIN0x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0

**Table 14-16. AESADIN Register Description**

Bit	Field	Type	Reset	Description
15-8	AESDIN1x	W	0h	AES data in byte n+1 when AESADIN is written as half-word. Do not use these bits for byte access. Do not mix half-word and byte access. Always reads as zero.
7-0	AESDIN0x	W	0h	AES data in byte n when AESADIN is written as half-word. AES next data in byte when AESADIN_L is written as byte. Do not mix half-word and byte access. Always reads as zero.

#### 14.3.6 AESADOUT Register

AES Accelerator Data Out Register

**Figure 14-19. AESADOUT Register**

15	14	13	12	11	10	9	8
AESDOUT1x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
AESDOUT0x							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

**Table 14-17. AESADOUT Register Description**

Bit	Field	Type	Reset	Description
15-8	AESDOUT1x	R	0h	AES data out byte n+1 when AESADOUT is read as half-word. Do not use these bits for byte access. Do not mix half-word and byte access.
7-0	AESDOUT0x	R	0h	AES data out byte n when AESADOUT is read as half-word. AES next data out byte when AESADOUT_L is read as byte. Do not mix half-word and byte access.

#### **14.3.7 AESAXDIN Register**

AES Accelerator XORed Data In Register

**Figure 14-20. AESAXDIN Register**

15	14	13	12	11	10	9	8
AESXDIN1x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0
7	6	5	4	3	2	1	0
AESXDIN0x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0

**Table 14-18. AESAXDIN Register Description**

Bit	Field	Type	Reset	Description
15-8	AESXDIN1x	W	0h	AES data in byte n+1 when AESAXDIN is written as half-word. Do not use these bits for byte access. Do not mix half-word and byte access. Always reads as zero.
7-0	AESXDIN0x	W	0h	AES data in byte n when AESAXDIN is written as half-word. AES next data in byte when AESAXDIN_L is written as byte. Do not mix half-word and byte access. Always reads as zero.

#### 14.3.8 AESAXIN Register

AES Accelerator XORed Data In Register (No Trigger)

**Figure 14-21. AESAXIN Register**

15	14	13	12	11	10	9	8
AESXIN1x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0
7	6	5	4	3	2	1	0
AESXIN0x							
w-0	w-0	w-0	w-0	w-0	w-0	w-0	w-0

**Table 14-19. AESAXIN Register Description**

Bit	Field	Type	Reset	Description
15-8	AESXIN1x	W	0h	AES data in byte n+1 when AESAXIN is written as half-word. Do not use these bits for byte access. Do not mix half-word and byte access. Always reads as zero.
7-0	AESXIN0x	W	0h	AES data in byte n when AESAXIN is written as half-word. AES next data in byte when AESAXIN_L is written as byte. Do not mix half-word and byte access. Always reads as zero.

## **Watchdog Timer (WDT\_A)**

The watchdog timer is a 32-bit timer that can be used as a watchdog or as an interval timer. This chapter describes the watchdog timer and its operating modes. The watchdog timer is implemented in all devices.

Topic	Page
15.1 WDT_A Introduction.....	578
15.2 WDT_A Operation.....	580
15.3 WDT_A Registers .....	583

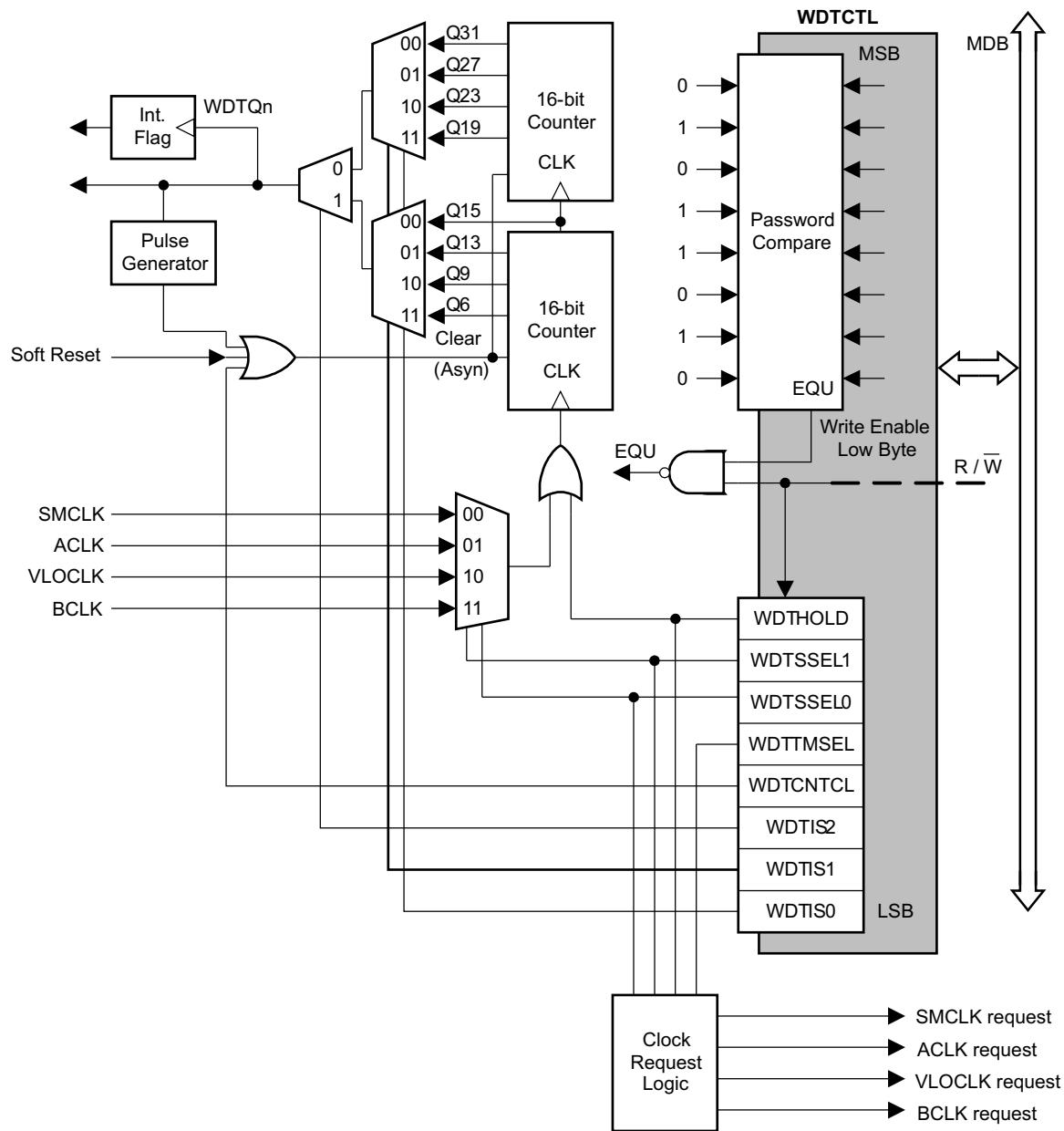
## 15.1 WDT\_A Introduction

The primary function of the watchdog timer (WDT\_A) module is to perform a controlled system restart after a software problem occurs. If the selected time interval expires, a system reset is generated. If the watchdog function is not needed in an application, the module can be configured as an interval timer and can generate interrupts at selected time intervals.

Features of the watchdog timer module include:

- Eight software-selectable time intervals
- Watchdog mode
- Interval timer mode
- Password-protected access to the Watchdog Timer Control (WDTCTL) register
- Selectable clock source
- Can be stopped to conserve power

[Figure 15-1](#) shows the watchdog timer block diagram.



**Figure 15-1. Watchdog Timer Block Diagram**

## 15.2 WDT\_A Operation

The watchdog timer module can be configured with the WDTCTL register as either a watchdog or an interval timer. WDTCTL is a 16-bit password-protected read/write register. Any read or write access must use half-word instructions, and write accesses must include the write password 05Ah in the upper byte. A write to WDTCTL with any value other than 05Ah in the upper byte is a password violation and causes a system reset, regardless of the WDT mode of operation. Any read of WDTCTL reads 069h in the upper byte. Writing byte wide only to the upper or lower parts of WDTCTL results in a system reset, as this particular register must always be accessed in half-word mode.

---

**NOTE: Watchdog timer powers up active.**

After a system reset, the WDT\_A module is automatically configured in the watchdog mode with a count value of  $2^{15}$ , using the SMCLK as the source. The application must set up or halt the WDT before the initial reset interval expires. As an example, if the SMCLK is default sourced by the DCO, which is set to 3 MHz, this results in an approximate 10.92-ms watchdog interval window. If the DCO frequency is changed, the application must control the watchdog time-out within the modified interval (see [Section 15.3.1](#)).

---

### 15.2.1 Watchdog Timer Counter (WDTCNT)

The WDTCNT is a 32-bit up counter that is not directly accessible by software. The WDTCNT is controlled and its time intervals are selected through the Watchdog Timer Control (WDTCTL) register. The WDTCNT can be sourced from SMCLK, ACLK, VLOCLK, and BCLK. The clock source is selected with the WDTSEL bits. The timer interval is selected with the WDTIS bits. This counter is automatically reset on a Soft Reset (or higher class of reset).

---

**NOTE:** The WDT counter is automatically configured to stop counting when the CPU is halted. This is to enable code development and debug without having to explicitly disable the WDT or without having to constantly encounter a watchdog initiated reset if the counter was allowed to continue running with the CPU halted. The application can choose to ignore the halt condition of the CPU. Refer to the SYSCTL chapter for more details on how this can be configured.

---

### 15.2.2 Watchdog Mode

After a system reset condition, the WDT\_A module is configured in the watchdog mode with an initial 10.92-ms (approximate) reset interval using the SMCLK. The user must set up, halt, or clear the watchdog timer before this initial reset interval expires, or another system reset is generated. When the watchdog timer is configured to operate in watchdog mode, either writing to WDTCTL with an incorrect password or expiration of the selected time interval also triggers a system reset. A system reset resets the watchdog timer to its default condition.

### 15.2.3 Interval Timer Mode

Setting the WDTTMSEL bit to 1 selects the interval timer mode. This mode can be used to provide periodic interrupts. In interval timer mode, the WDT generates an interrupt at the end of each interval. Refer to [Section 15.2.4](#) for details on handling of WDT related interrupt flags.

---

**NOTE: Modifying the watchdog timer**

The watchdog timer interval should be changed together with WDTCNTCL = 1 in a single instruction to avoid an unexpected immediate system reset or interrupt. The watchdog timer should be halted before changing the clock source to avoid a possible incorrect interval.

---

### 15.2.4 Watchdog Related Interrupts and Flags

The watchdog timer interrupt is handled through the enable, set, clear, pending flag of the Cortex-M4 NVIC channel that the WDT is mapped to. Refer to the appropriate device data sheet for the NVIC channel number where the WDT is mapped. The functionality of the NVIC ensures that the related interrupt pending flag is automatically cleared when the interrupt is serviced (unless another interrupt occurs before the ISR is completed, which causes a re-entry into the same ISR).

When using the watchdog timer in the watchdog mode, a reset issued by the WDT may be mapped either as the Hard Reset or the Soft Reset of the device. In addition, flags in the Reset Controller can be used to determine if the WDT was indeed the cause of the reset. Refer to the appropriate device data sheet to determine the reset channel of the Reset Controller that the WDT reset is mapped to. In addition, refer to the *Reset Controller* chapter for details on how the flags of the various reset sources can be processed by the application. These flags in the Reset Controller can be used by the reset service routine to determine if the watchdog caused the device to reset. If the flag is set, the watchdog timer initiated the reset condition, either by timing out or by a password violation. If the WDT flag is not set, the reset was caused by a different source.

### 15.2.5 Clock Sources of the WDT\_A

The WDT\_A provides multiple options for the clock that can be used to source the counter, either in watchdog or in interval timer mode (see [Table 15-1](#)). The appropriate source can be selected by controlling the WDTSEL bits in the WDTCTL register.

**Table 15-1. WDT\_A Clock Sources**

WDTSELx	Clock Source Selected (Watchdog and Interval Timer Mode) <sup>(1)</sup> , <sup>(2)</sup>
00	SMCLK
01	ACLK
10	VLOCLK
11	BCLK

<sup>(1)</sup> Refer to the CS chapter for more details on the SMCLK, ACLK, VLOCLK and BCLK settings as well as the possible clock sources for the same.

<sup>(2)</sup> Refer to [Section 15.2.6](#) for details on clock source options during the low-power modes of the device.

#### 15.2.5.1 Clock Fail-Safe Feature

If any of the sources for the clocks in [Table 15-1](#) fails, the Clock System (CS) automatically switches over to the appropriate failsafe option, so that the WDT operation can continue. Refer to the *Clock System (CS)* chapter for more details on the failsafe options for the various clock sources on the device.

### 15.2.6 WDT\_A Operation in Different Device Power Modes

The devices have several active and low-power modes of operation. The requirements of the application and the type of clocking that is used determine how the WDT\_A should be configured.

#### 15.2.6.1 WDT\_A Operation in Active Modes

The WDT\_A may be used either in watchdog or interval timer mode when the device is in one of the active modes of operation. All clock sources as listed in [Table 15-1](#) are available.

#### 15.2.6.2 WDT\_A Operation in LPM0 Modes

The WDT\_A may be used either in watchdog or interval timer mode when the device is in one of the LPM0 modes of operation. All clock sources as listed in [Table 15-1](#) are available.

#### 15.2.6.3 WDT\_A Operation in LPM3 Mode

The WDT\_A remains functional in the LPM3 mode of operation. However, it may be used **only in interval timer mode**. This is because there is no execution activity to handle the watchdog, and a reset during a LPM3 condition may result in a nondeterministic device state when it returns to active mode. On the other hand, if the WDT\_A is configured as an interval timer, the interval timer event can be used to wake up the device back to active mode with full state retention, and the interval timer interrupt is processed correctly.

Before invoking the device entry to LPM3 mode, the WDT\_A must be configured to use either BCLK or VLOCLK as the source. If any of the other clock sources are used, the WDT\_A may prevent the device from entering LPM3 mode and the device will consume excess current as a result. Refer to the *Power Control Manager (PCM)* chapter for more details about the clock handling during entry to low-power modes.

#### 15.2.6.4 WDT\_A Operation in LPM3.5 Mode

The WDT\_A remains functional in the LPM3.5 mode of operation. However, it may be used **only in interval timer mode**. This is because there is no execution activity to handle the watchdog. On the other hand, if the WDT\_A is configured as an interval timer, the interval timer event can be used to wake up the device back to active mode.

Before invoking the device entry to LPM3.5 mode, the WDT\_A must be configured to use either BCLK or VLOCLK as the source. If any of the other clock sources are used, the WDT\_A may prevent the device from entering LPM3.5 mode and the device will consume excess current as a result. Refer to the *Power Control Manager (PCM)* chapter for more details about the clock handling during entry to low-power modes.

Upon exit from LPM3.5 mode to active mode, the device undergoes a reset, but the WDT\_A remains functional and unaffected by the reset. This isolation is carried out through the LOCKBKUP and LOCKLPM5 bits in the PCM. Refer to the PCM chapter for more details on entering and exiting LPM3.5 mode of operation.

---

**NOTE:** While the WDT\_A operation remains unaffected during and after the LPM3.5 mode, the WDTCTL register is reset due to the subsequent device reset. It is the responsibility of the application to re-initialize the WDTCTL register to the value prior to LPM3.5 mode entry before clearing the LOCKBKUP and LOCKLPM5 bits, else the WDT\_A operation is affected by the reset settings of the register.

---

#### 15.2.6.5 WDT\_A Operation in LPM4 and LPM4.5 Modes

The WDT\_A functionality is not available in LPM4 and LPM4.5 modes.

## 15.3 WDT\_A Registers

Table 15-2 shows the WDT\_A module registers and their address offsets. The base address can be found in the device-specific data sheet.

**Table 15-2. WDT\_A Registers**

Offset	Acronym	Register Name	Section
0Ch	WDTCTL	Watchdog Timer Control	Section 15.3.1

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through half-word (16 bit) access.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 15.3.1 WDTCTL Register

Watchdog Timer Control Register

**Figure 15-2. WDTCTL Register**

15	14	13	12	11	10	9	8
WDTPW							
rw-{0}	rw-{1}	rw-{1}	rw-{0}	rw-{1}	rw-{0}	rw-{0}	rw-{1}
7	6	5	4	3	2	1	0
WDTHOLD	WDTSEL	WDTMSEL	WDTCNTCL		WDTIS		
rw-{0}	rw-{0}	rw-{0}	rw-{0}	rw-{0}	rw-{1}	rw-{0}	rw-{0}

**Table 15-3. WDTCTL Register Description<sup>(1)</sup>**

Bit	Field	Type	Reset	Description
15-8	WDTPW	RW	69h	Watchdog timer password. Always read as 069h. Must be written as 05Ah, or the WDT generates a reset.
7	WDTHOLD	RW	0h	Watchdog timer hold. This bit stops the watchdog timer. Setting WDTHOLD = 1 when the WDT is not in use conserves power. 0b = Watchdog timer is not stopped 1b = Watchdog timer is stopped
6-5	WDTSEL	RW	0h	Watchdog timer clock source select 00b = SMCLK 01b = ACLK 10b = VLOCLK 11b = BCLK
4	WDTMSEL	RW	0h	Watchdog timer mode select 0b = Watchdog mode <sup>(2)</sup> 1b = Interval timer mode <sup>(3)</sup>
3	WDTCNTCL	W	0h	Watchdog timer counter clear. Setting WDTCNTCL = 1 clears the count value to 0000h. This bit always reads 0h 0b = No action 1b = WDTCNT = 0000h
2-0	WDTIS	RW	4h	Watchdog timer interval select. These bits select the watchdog timer interval to generate either a WDT interrupt or a WDT reset. 000b = Watchdog clock source / $2^{31}$ (18:12:16 at 32.768 kHz) 001b = Watchdog clock source / $2^{27}$ (01:08:16 at 32.768 kHz) 010b = Watchdog clock source / $2^{23}$ (00:04:16 at 32.768 kHz) 011b = Watchdog clock source / $2^{19}$ (00:00:16 at 32.768 kHz) 100b = Watchdog clock source / $2^{15}$ (1 s at 32.768 kHz) 101b = Watchdog clock source / $2^{13}$ (250 ms at 32.768 kHz) 110b = Watchdog clock source / $2^9$ (15.625 ms at 32.768 kHz) 111b = Watchdog clock source / $2^6$ (1.95 ms at 32.768 kHz)

<sup>(1)</sup> Writes to this register must be 16 bits wide; otherwise, the WDT\_A generates a reset.

<sup>(2)</sup> Clock requests to ACLK and SMCLK clocks in watchdog mode of operation are unconditional clock requests. So disabling the corresponding clock enable bits in CS does not stop the clock to the WDT\_A module.

<sup>(3)</sup> In the Interval time mode, ACLK and SMCLK clock requests are always conditional from WDT\_A module.

**Timer32**

This chapter describes the features and functionality of Timer32.

Topic	Page
<b>16.1 Introduction .....</b>	<b>586</b>
<b>16.2 Functional Description .....</b>	<b>586</b>
<b>16.3 Operation .....</b>	<b>586</b>
<b>16.4 Interrupt Generation.....</b>	<b>587</b>
<b>16.5 Timer32 Registers.....</b>	<b>588</b>

## 16.1 Introduction

The Timer32 is an AMBA compliant peripheral developed, tested, and licensed by ARM Limited. The Timer32 consists of two programmable 32-bit or 16-bit down counters that can generate interrupts on reaching zero.

The key features of Timer32 include:

- Two independent counters each configurable as 32-bit or 16-bit counter size
- Three different timer modes supported for each counter
- Prescale unit to divide the input clock by 1, 16 or 256
- Independent Interrupts from each of the counter, as well as, a combined interrupt from both the counters

## 16.2 Functional Description

Two independent timers are available in Timer32. For each of the timers, the following modes of operation are available:

- Free-running mode: The counter wraps after reaching its zero value, and continues to count down from the maximum value. This is the default mode.
- Periodic timer mode: The counter generates an interrupt at a constant interval, reloading the original value after wrapping past zero.
- One-shot timer mode: The counter generates an interrupt once. When the counter reaches zero, it halts until reprogrammed by the user. This can be achieved by either clearing the One Shot Count bit in the control register, in which case the count proceeds according to the selection of free-running or periodic mode, or by writing a new value to the Load Value register.

## 16.3 Operation

Each timer has an identical set of registers and the operation of each timer is also identical. The timer is loaded by writing to the load register and, if enabled, counts down to zero. When a counter is already running, writing to the load register causes the counter to immediately restart at the new value. Writing to the background load value has no effect on the current count. The counter continues to decrement to zero, and then recommences from the new load value, if in periodic mode, and one shot mode is not selected.

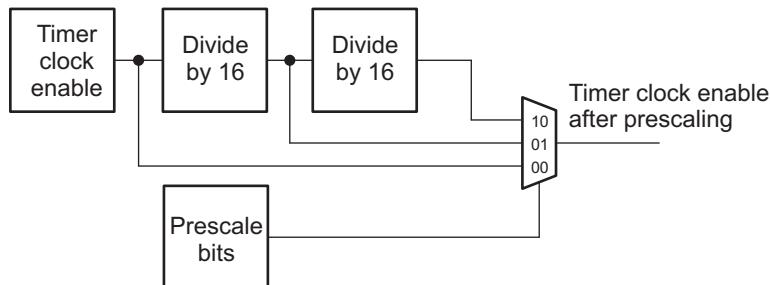
When zero is reached, an interrupt is generated. The interrupt can be cleared by writing to the clear register. If One Shot Mode is selected, the counter halts on reaching zero until One Shot Mode is deselected, or a new Load value is written. Otherwise, after reaching a zero count, if the timer is operating in free-running mode it continues to decrement from its maximum value. If periodic timer mode is selected, the timer reloads the count value from the Load Register and continues to decrement. In this mode the counter effectively generates a periodic interrupt. The mode is selected by a bit in the Timer Control Register. At any point, the current counter value can be read from the Current Value Register. The counter is enabled by the ENABLE bit in the T32CONTROLx register.

At reset, the counter is disabled, the interrupt is cleared, and the load register is set to zero. The mode and prescale values are set to free-running, and clock divide of 1 respectively.

The timer clock enable is generated by a prescale unit. The enable is then used by the counter to create a clock with a timing of one of the following:

- MCLK
- MCLK divided by 16, generated by 4 bits of prescale
- MCLK divided by 256, generated by a total of 8 bits of prescale

[Figure 16-1](#) shows how the timer clock frequency is selected in the prescale unit. This enables the timer to be clocked at different frequencies.



**Figure 16-1. Prescale Clock Enable Generation**

## 16.4 Interrupt Generation

An interrupt is generated when the full 32-bit counter reaches zero and is only cleared when the T32INTCLR<sub>x</sub> register is written to. A register holds the value until the interrupt is cleared. The most significant carry bit of the counter detects the counter reaching zero.

The interrupts can be masked by writing 0 to the Interrupt Enable bit in the T32CONTROL<sub>x</sub> register. Both the raw interrupt status, prior to masking, and the final interrupt status, after masking, can be read from status registers. The interrupts from the individual counters, after masking, are logically ORed into a combined interrupt, TIMINTC, provides an additional interrupt condition from the Timer32 peripheral. Thus, the module supports three interrupts in total – TIMINT1, TIMINT2, and TIMINTC.

## 16.5 Timer32 Registers

[Table 16-1](#) lists the Timer32 registers and their address offsets. See the device-specific data sheet for the base address of the module.

**Table 16-1. Timer32 Registers**

Offset	Acronym	Register Name	Type	Reset	Section
00h	T32LOAD1	Timer 1 Load Register	RW	0h	<a href="#">Section 16.5.1</a>
04h	T32VALUE1	Timer 1 Current Value Register	R	FFFFFFFh	<a href="#">Section 16.5.2</a>
08h	T32CONTROL1	Timer 1 Timer Control Register	RW	20h	<a href="#">Section 16.5.3</a>
0Ch	T32INTCLR1	Timer 1 Interrupt Clear Register	W	-	<a href="#">Section 16.5.4</a>
10h	T32RIS1	Timer 1 Raw Interrupt Status Register	R	0h	<a href="#">Section 16.5.5</a>
14h	T32MIS1	Timer 1 Interrupt Status Register	R	0h	<a href="#">Section 16.5.6</a>
18h	T32BGLOAD1	Timer 1 Background Load Register	RW	0h	<a href="#">Section 16.5.7</a>
20h	T32LOAD2	Timer 2 Load Register	RW	0h	<a href="#">Section 16.5.8</a>
24h	T32VALUE2	Timer 2 Current Value Register	R	FFFFFFFh	<a href="#">Section 16.5.9</a>
28h	T32CONTROL2	Timer 2 Timer Control Register	RW	20h	<a href="#">Section 16.5.10</a>
2Ch	T32INTCLR2	Timer 2 Interrupt Clear Register	W	X	<a href="#">Section 16.5.11</a>
30h	T32RIS2	Timer 2 Raw Interrupt Status Register	R	0h	<a href="#">Section 16.5.12</a>
34h	T32MIS2	Timer 2 Interrupt Status Register	R	0h	<a href="#">Section 16.5.13</a>
38h	T32BGLOAD2	Timer 2 Background Load Register	RW	0h	<a href="#">Section 16.5.14</a>

---

**NOTE:** This is a 32-bit module and can be accessed ONLY through word (32-bit) access.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 16.5.1 T32LOAD1 Register (offset = 00h) [reset = 0h]

#### Timer1 Load Register

The T32LOAD1 register is a 32-bit register containing the value from which the counter is to decrement. This is the value used to reload the counter when Periodic mode is enabled, and the current count reaches zero. When this register is written to directly, the current count is immediately reset to the new value at the next rising edge of TIMCLK. The value in this register is also overwritten if the T32BGLOAD1 register is written to, but the current count is not immediately affected.

If values are written to both the T32LOAD1 and T32BGLOAD1 registers before an enabled rising edge on TIMCLK, the following occurs:

- On the next enabled TIMCLK edge, the value written to the T32LOAD1 value replaces the current count value.
- Then, each time the counter reaches zero, the current count value is reset to the value written to T32BGLOAD1

Reading from the T32LOAD1 register at any time after the two writes have occurred retrieves the value written to T32BGLOAD1. That is, the value read from T32LOAD1 is always the value that takes effect for Periodic mode after the next time the counter reaches zero.

**Figure 16-2. T32LOAD1 Register**

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
LOAD																															
rw-0																															

**Table 16-2. T32LOAD1 Register Description**

Bit	Field	Type	Reset	Description
31-0	LOAD	RW	0h	The value from which the Timer 1 counter decrements

### 16.5.2 T32VALUE1 Register (offset = 04h) [reset = FFFFFFFFh]

This register contains Timer 1 Current Value

**Figure 16-3. T32VALUE1 Register**

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
VALUE																															
r-FFFFFFFh																															

**Table 16-3. T32VALUE1 Register Description**

Bit	Field	Type	Reset	Description
31-0	VALUE	R	FFFFFFFh	Reports the current value of the decrementing counter

### 16.5.3 T32CONTROL1 Register (offset = 08h) [reset = 20h]

Timer 1 Timer Control Register

**Figure 16-4. T32CONTROL1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ENAB LE	MODE	IE	Reserv ed	PRESCALE	SIZE	ONES HOT	
r-0								rw-0	rw-0	rw-1	r-0	rw-0	rw-0	rw-0	

**Table 16-4. T32CONTROL1 Register Description**

Bit	Field	Type	Reset	Description
31-8	Reserved	R	0h	Reserved
7	ENABLE	RW	0h	Enable bit 0b = Timer disabled 1b = Timer enabled
6	MODE	RW	0h	Mode bit 0b = Timer is in free-running mode 1b = Timer is in periodic mode
5	IE	RW	1h	Interrupt enable bit 0b = Timer interrupt disabled 1b = Timer interrupt enabled
4	Reserved	R	0h	Reserved
3-2	PRESCALE	RW	0h	Prescale bits 00b = 0 stages of prescale, clock is divided by 1 01b = 4 stages of prescale, clock is divided by 16 10b = 8 stages of prescale, clock is divided by 256 11b = Reserved
1	SIZE	RW	0h	Selects 16 or 32 bit counter operation 0b = 16-bit counter 1b = 32-bit counter
0	ONESHOT	RW	0h	Selects one-shot or wrapping counter mode 0b = wrapping mode 1b = one-shot mode

#### **16.5.4 T32INTCLR1 Register (offset = 0Ch) [reset = undefined]**

Timer 1 Interrupt Clear Register

**Figure 16-5. T32INTCLR1 Register**

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
INTCLR																															
W																															

**Table 16-5. T32INTCLR1 Register Description**

Bit	Field	Type	Reset	Description
31-0	INTCLR	W	x	Any write to the T32INTCLR1 register clears the interrupt output from the counter.

### **16.5.5 T32RIS1 Register (offset = 10h) [reset = 0h]**

#### Timer 1 Raw Interrupt Status Register

This register indicates the raw interrupt status from the counter. This value is combined by a logical AND with the timer interrupt enable bit from the Timer Control Register to create the masked interrupt, which is passed to the interrupt output pin.

**Figure 16-6. T32RIS1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RAW_I FG
r-0															r-0

**Table 16-6. T32RIS1 Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved
0	RAW_IFG	R	0h	Raw interrupt status from the counter

### 16.5.6 T32MIS1 Register (offset = 14h) [reset = 0h]

#### Timer 1 Masked Interrupt Status Register

This register indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the timer interrupt enable bit from the Timer Control Register. This is the same value that is passed to the interrupt output pin.

**Figure 16-7. T32MIS1 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															IFG
r-0															

**Table 16-7. T32MIS1 Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved
0	IFG	R	0h	Enabled interrupt status from the counter

### 16.5.7 T32BGLOAD1 Register (offset = 18h) [reset = 0h]

Timer 1 Background Load Register

The T32BGLOAD1 register is 32-bits and contains the value from which the counter is to decrement. This is the value used to reload the counter when Periodic mode is enabled, and the current count reaches zero.

This register provides an alternative method of accessing the T32LOAD1 Register. The difference is that writes to T32BGLOAD1 do not cause the counter to immediately restart from the new value.

Reading from this register returns the same value returned from T32LOAD1. See [Section 16.5.1](#) for more details.

**Figure 16-8. T32BGLOAD1 Register**

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
BGLOAD																															
rw-0																															

**Table 16-8. T32BGLOAD1 Register Description**

Bit	Field	Type	Reset	Description
31-0	BGLOAD	RW	0h	Contains the value from which the counter decrements

### 16.5.8 T32LOAD2 Register (offset = 20h) [reset = 0h]

#### Timer2 Load Register

The T32LOAD2 register is a 32-bit register containing the value from which the counter is to decrement. This is the value used to reload the counter when Periodic mode is enabled, and the current count reaches zero. When this register is written to directly, the current count is immediately reset to the new value at the next rising edge of TIMCLK. The value in this register is also overwritten if the T32BGLOAD2 register is written to, but the current count is not immediately affected.

If values are written to both the T32LOAD2 and T32BGLOAD2 registers before an enabled rising edge on TIMCLK, the following occurs:

- On the next enabled TIMCLK edge, the value written to the T32LOAD2 value replaces the current count value.
- Then, each time the counter reaches zero, the current count value is reset to the value written to T32BGLOAD2

Reading from the T32LOAD2 register at any time after the two writes have occurred retrieves the value written to T32BGLOAD2. That is, the value read from T32LOAD2 is always the value that takes effect for Periodic mode after the next time the counter reaches zero.

**Figure 16-9. T32LOAD2 Register**

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
LOAD																															
rw-0																															

**Table 16-9. T32LOAD2 Register Description**

Bit	Field	Type	Reset	Description
31-0	LOAD	RW	0h	The value from which the Timer 2 counter decrements

**16.5.9 T32VALUE2 Register (offset = 24h) [reset = FFFFFFFFh]**

This register contains Timer 2 Current Value

**Figure 16-10. T32VALUE2 Register**

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
VALUE																															
r-FFFFFFFh																															

**Table 16-10. T32VALUE2 Register Description**

Bit	Field	Type	Reset	Description
31-0	VALUE	R	FFFFFFFh	Reports the current value of the decrementing counter

**16.5.10 T32CONTROL2 Register (offset = 28h) [reset = 20h]**

Timer 2 Timer Control Register

**Figure 16-11. T32CONTROL2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ENAB LE	MODE	IE	Reserv ed	PRESCALE	SIZE	ONES HOT	
r-0								rw-0	rw-0	rw-1	r-0	rw-0	rw-0	rw-0	

**Table 16-11. T32CONTROL2 Register Description**

Bit	Field	Type	Reset	Description
31-8	Reserved	R	0h	Reserved
7	ENABLE	RW	0h	Enable bit 0b = Timer disabled 1b = Timer enabled
6	MODE	RW	0h	Mode bit 0b = Timer is in free-running mode 1b = Timer is in periodic mode
5	IE	RW	1h	Interrupt enable bit 0b = Timer interrupt disabled 1b = Timer interrupt enabled
4	Reserved	R	0h	Reserved
3-2	PRESCALE	RW	0h	Prescale bits 00b = 0 stages of prescale, clock is divided by 1 01b = 4 stages of prescale, clock is divided by 16 10b = 8 stages of prescale, clock is divided by 256 11b = Reserved
1	SIZE	RW	0h	Selects 16 or 32 bit counter operation 0b = 16-bit counter 1b = 32-bit counter
0	ONESHOT	RW	0h	Selects one-shot or wrapping counter mode 0b = wrapping mode 1b = one-shot mode

**16.5.11 T32INTCLR2 Register (offset = 2Ch) [reset = undefined]**

Timer 2 Interrupt Clear Register

**Figure 16-12. T32INTCLR2 Register**

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
INTCLR																															
W																															

**Table 16-12. T32INTCLR2 Register Description**

Bit	Field	Type	Reset	Description
31-0	INTCLR	W	x	Any write to the T32INTCLR2 register clears the interrupt output from the counter.

### 16.5.12 T32RIS2 Register (offset = 30h) [reset = 0h]

Timer 2 Raw Interrupt Status Register

This register indicates the raw interrupt status from the counter. This value is combined by a logical AND with the timer interrupt enable bit from the Timer Control Register to create the masked interrupt, which is passed to the interrupt output pin.

**Figure 16-13. T32RIS2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RAW_I FG
r-0															r-0

**Table 16-13. T32RIS2 Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved
0	RAW_IFG	R	0h	Raw interrupt status from the counter

### **16.5.13 T32MIS2 Register (offset = 34h) [reset = 0h]**

Timer 2 Masked Interrupt Status Register

This register indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the timer interrupt enable bit from the Timer Control Register. This is the same value that is passed to the interrupt output pin.

**Figure 16-14. T32MIS2 Register**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															IFG
r-0															

**Table 16-14. T32MIS2 Register Description**

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	Reserved
0	IFG	R	0h	Enabled interrupt status from the counter

### 16.5.14 T32BGLOAD2 Register (offset = 38h) [reset = 0h]

Timer 2 Background Load Register

The T32BGLOAD2 register is 32-bits and contains the value from which the counter is to decrement. This is the value used to reload the counter when Periodic mode is enabled, and the current count reaches zero.

This register provides an alternative method of accessing the T32LOAD2 Register. The difference is that writes to T32BGLOAD2 do not cause the counter to immediately restart from the new value.

Reading from this register returns the same value returned from T32LOAD2. See [Section 16.5.8](#) for more details.

**Figure 16-15. T32BGLOAD2 Register**

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
BGLOAD																															
rw-0																															

**Table 16-15. T32BGLOAD2 Register Description**

Bit	Field	Type	Reset	Description
31-0	BGLOAD	RW	0h	Contains the value from which the counter decrements

***Timer\_A***

Timer\_A is a 16-bit timer/counter with multiple capture/compare registers. There can be multiple Timer\_A modules on a given device (see the device-specific data sheet). This chapter describes the operation and use of the Timer\_A module.

Topic	Page
17.1 Timer_A Introduction .....	604
17.2 Timer_A Operation.....	606
17.3 Timer_A Registers .....	617

## 17.1 Timer\_A Introduction

Timer\_A is a 16-bit timer/counter with up to seven capture/compare registers. Timer\_A can support multiple capture/comparisons, PWM outputs, and interval timing. Timer\_A also has extensive interrupt capabilities. Interrupts may be generated from the counter on overflow conditions and from each of the capture/compare registers.

Timer\_A features include:

- Asynchronous 16-bit timer/counter with four operating modes
- Selectable and configurable clock source
- Up to seven configurable capture/compare registers
- Configurable outputs with pulse width modulation (PWM) capability
- Asynchronous input and output latching

The block diagram of Timer\_A is shown in [Figure 17-1](#).

---

**NOTE: Use of the word count**

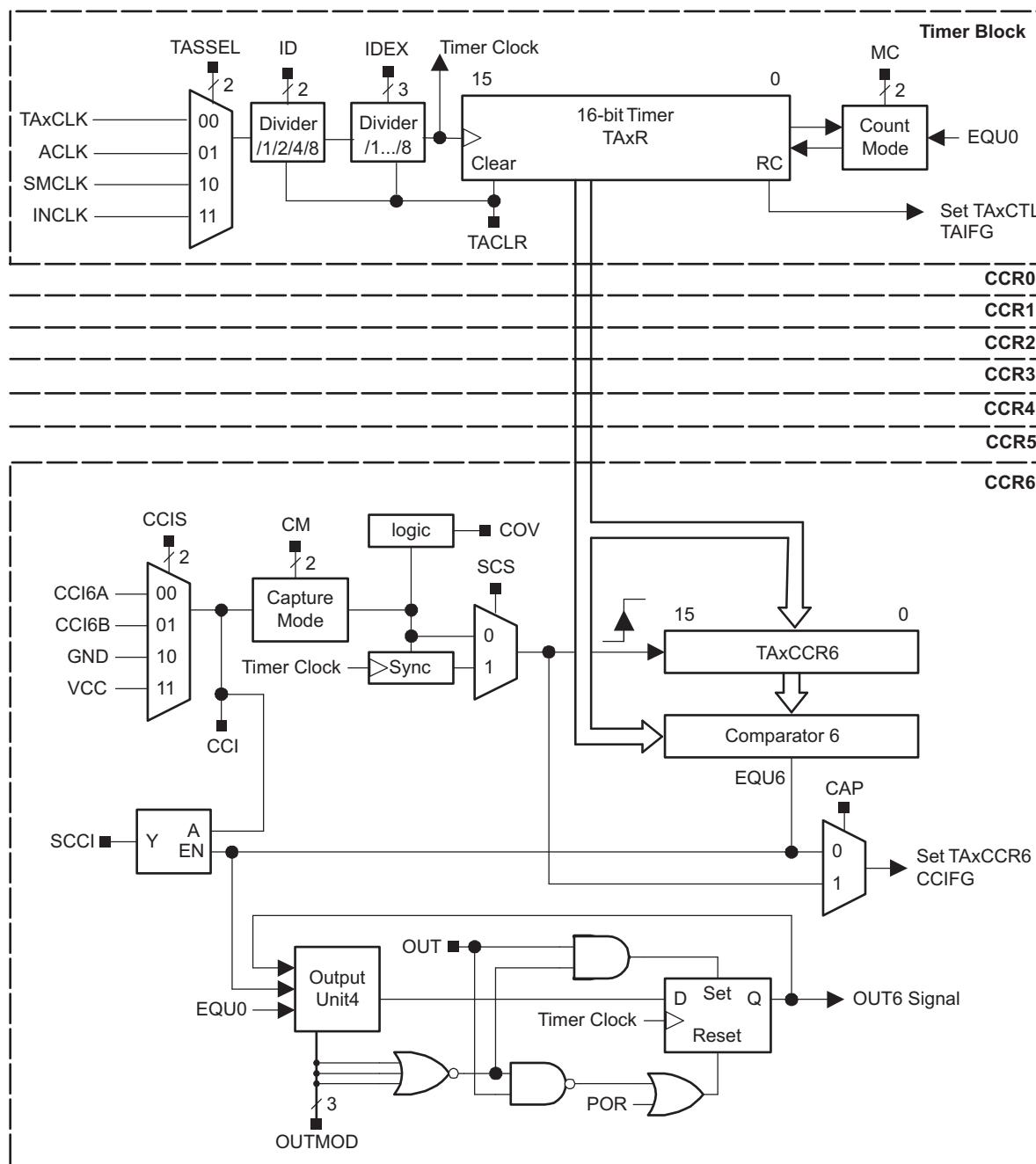
*Count* is used throughout this chapter. It means the counter must be in the process of counting for the action to take place. If a particular value is directly written to the counter, an associated action does not take place.

---

**NOTE: Nomenclature**

There may be multiple instantiations of Timer\_A on a given device. The prefix TAx is used, where x is a greater than equal to zero indicating the Timer\_A instantiation. For devices with one instantiation, x = 0. The suffix n, where n = 0 to 6, represents the specific capture/compare registers associated with the Timer\_A instantiation.

---



Copyright © 2016, Texas Instruments Incorporated

**Figure 17-1. Timer\_A Block Diagram**

## 17.2 Timer\_A Operation

The Timer\_A module is configured with user software. The setup and operation of Timer\_A are discussed in the following sections.

### 17.2.1 16-Bit Timer Counter

The 16-bit timer/counter register, TAxR, increments or decrements (depending on mode of operation) with each rising edge of the clock signal. TAxR can be read or written with software. Additionally, the timer can generate an interrupt when it overflows.

TAxR may be cleared by setting the TACLR bit. Setting TACLR also clears the clock divider and count direction for up/down mode.

---

**NOTE: Modifying Timer\_A registers**

It is recommended to stop the timer before modifying its operation (with exception of the interrupt enable, interrupt flag, and TACLR) to avoid errant operating conditions.

When the timer clock is asynchronous to the CPU clock, any read from TAxR should occur while the timer is not operating or the results may be unpredictable. Alternatively, the timer may be read multiple times while operating, and a majority vote taken in software to determine the correct reading. Any write to TAxR takes effect immediately.

---

#### 17.2.1.1 Clock Source Select and Divider

The timer clock can be sourced from ACLK, SMCLK, or externally from TAxCLK or INCLK. The clock source is selected with the TASSEL bits. The selected clock source may be passed directly to the timer or divided by 2, 4, or 8, using the ID bits. The selected clock source can be further divided by 2, 3, 4, 5, 6, 7, or 8 using the TAIDEX bits. The timer clock divider logic is reset when TACLR is set.

---

**NOTE: Timer\_A dividers**

After programming ID or TAIDEX bits, set the TACLR bit. This clears the contents of TAxR and resets the clock divider logic to a defined state. The clock dividers are implemented as down counters. Therefore, when the TACLR bit is cleared, the timer clock immediately begins clocking at the first rising edge of the Timer\_A clock source selected with the TASSEL bits and continues clocking at the divider settings set by the ID and TAIDEX bits.

---

---

**NOTE:** When the timer is clocked by an external clock, the first two clock pulses are missed due to internal clock synchronization scheme and the timer starts counting only from the third clock pulse.

### 17.2.2 Starting the Timer

The timer may be started or restarted in the following ways:

- The timer counts when MC > { 0 } and the clock source is active.
- When the timer mode is either up or up/down, the timer may be stopped by writing 0 to TAxCCR0. The timer may then be restarted by writing a nonzero value to TAxCCR0. In this scenario, the timer starts incrementing in the up direction from zero.

### 17.2.3 Timer Mode Control

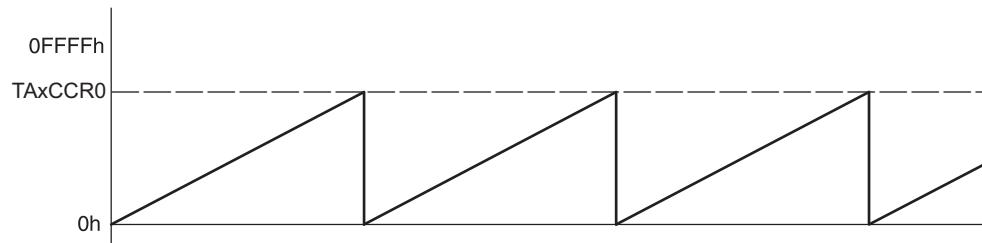
The timer has four modes of operation: stop, up, continuous, and up/down (see [Table 17-1](#)). The operating mode is selected with the MC bits.

**Table 17-1. Timer Modes**

MC	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of TAxCCR0
10	Continuous	The timer repeatedly counts from zero to 0FFFFh.
11	Up/down	The timer repeatedly counts from zero up to the value of TAxCCR0 and back down to zero.

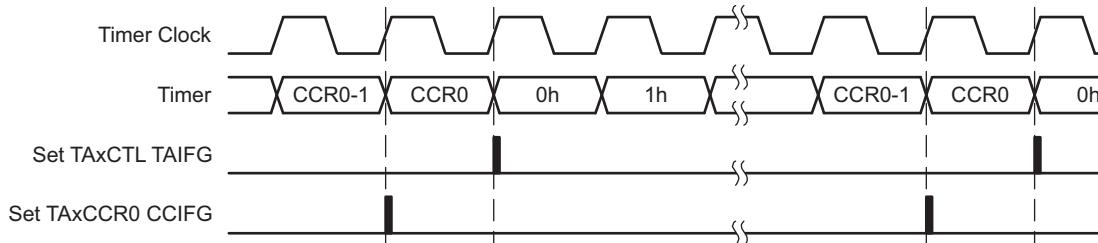
#### 17.2.3.1 Up Mode

The up mode is used if the timer period must be different from 0FFFFh counts. The timer repeatedly counts up to the value of compare register TAxCCR0, which defines the period (see [Figure 17-2](#)). The number of timer counts in the period is TAxCCR0 + 1. When the timer value equals TAxCCR0, the timer restarts counting from zero. If up mode is selected when the timer value is greater than TAxCCR0, the timer immediately restarts counting from zero.



**Figure 17-2. Up Mode**

The TAxCCR0 CCIFG interrupt flag is set when the timer *counts* to the TAxCCR0 value. The TAIFG interrupt flag is set when the timer *counts* from TAxCCR0 to zero. [Figure 17-3](#) shows the flag set cycle.



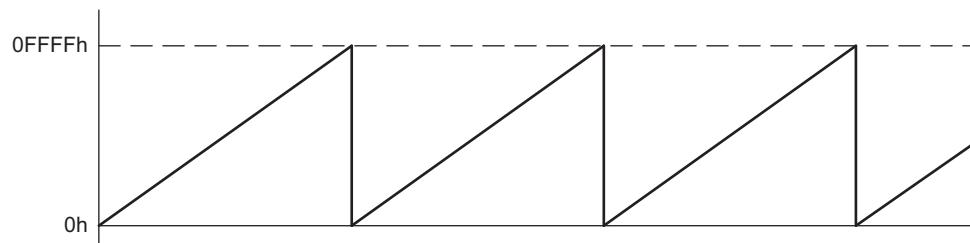
**Figure 17-3. Up Mode Flag Setting**

##### 17.2.3.1.1 Changing Period Register TAxCCR0

When changing TAxCCR0 while the timer is running, if the new period is greater than or equal to the old period or greater than the current count value, the timer counts up to the new period. If the new period is less than the current count value, the timer rolls to zero. However, one additional count may occur before the counter rolls to zero.

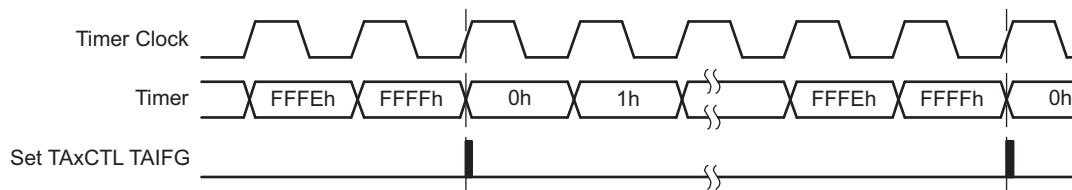
### 17.2.3.2 Continuous Mode

In the continuous mode, the timer repeatedly counts up to 0FFFFh and restarts from zero as shown in [Figure 17-4](#). The capture/compare register TAxCCR0 works the same way as the other capture/compare registers.



**Figure 17-4. Continuous Mode**

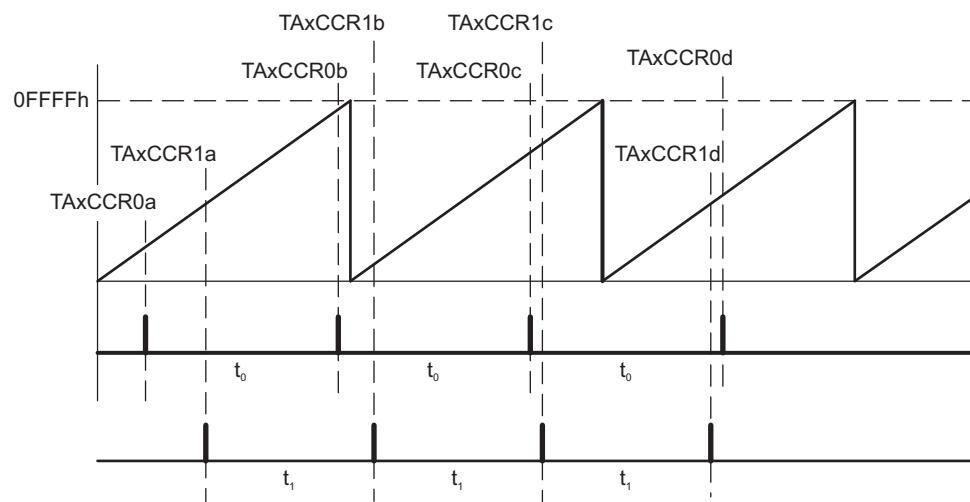
The TAIFG interrupt flag is set when the timer *counts* from 0FFFFh to zero. [Figure 17-5](#) shows the flag set cycle.



**Figure 17-5. Continuous Mode Flag Setting**

### 17.2.3.3 Use of Continuous Mode

The continuous mode can be used to generate independent time intervals and output frequencies. Each time an interval is completed, an interrupt is generated. The next time interval is added to the TAxCCRn register in the interrupt service routine. [Figure 17-6](#) shows two separate time intervals,  $t_0$  and  $t_1$ , being added to the capture/compare registers. In this usage, the time interval is controlled by hardware, not software, without impact from interrupt latency. Up to n (where n = 0 to 6), independent time intervals or output frequencies can be generated using capture/compare registers.

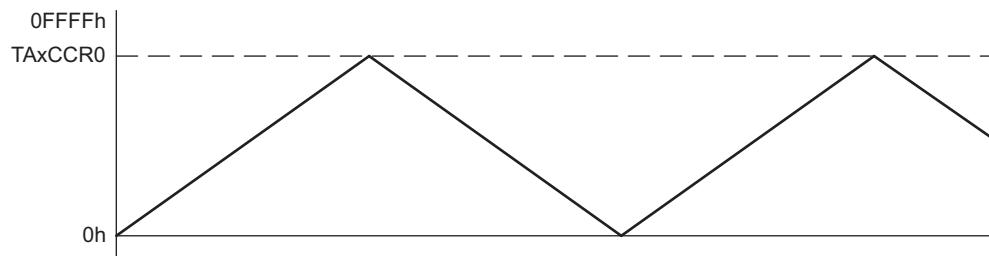


**Figure 17-6. Continuous Mode Time Intervals**

Time intervals can be produced with other modes as well, where TAxCCR0 is used as the period register. Their handling is more complex since the sum of the old TAxCCRn data and the new period can be higher than the TAxCCR0 value. When the previous TAxCCRn value plus  $t_x$  is greater than the TAxCCR0 data, the TAxCCR0 value must be subtracted to obtain the correct time interval.

#### 17.2.3.4 Up/Down Mode

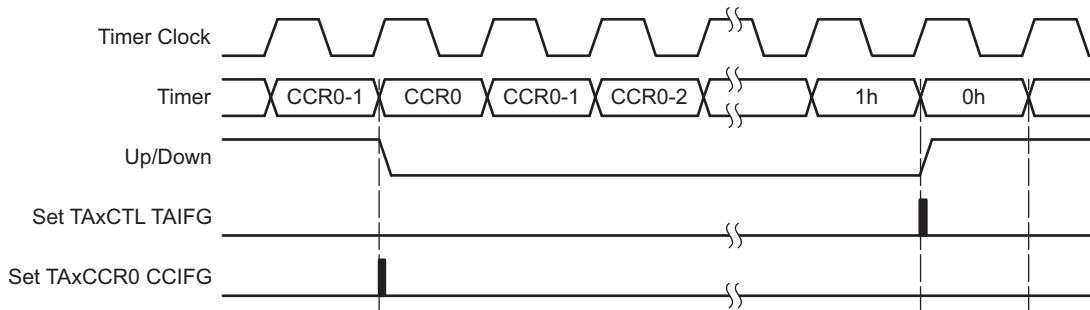
The up/down mode is used if the timer period must be different from 0FFFFh counts, and if symmetrical pulse generation is needed. The timer repeatedly counts up to the value of compare register TAxCCR0 and back down to zero (see [Figure 17-7](#)). The period is twice the value in TAxCCR0.



**Figure 17-7. Up/Down Mode**

The count direction is latched. This allows the timer to be stopped and then restarted in the same direction it was counting before it was stopped. If this is not desired, the TACLR bit must be set to clear the direction. The TACLR bit also clears the TAxR value and the timer clock divider.

In up/down mode, the TAxCCR0 CCIFG interrupt flag and the TAIFG interrupt flag are set only once during a period, separated by one-half the timer period. The TAxCCR0 CCIFG interrupt flag is set when the timer *counts* from TAxCCR0-1 to TAxCCR0, and TAIFG is set when the timer completes *counting* down from 0001h to 0000h. [Figure 17-8](#) shows the flag set cycle.



**Figure 17-8. Up/Down Mode Flag Setting**

##### 17.2.3.4.1 Changing Period Register TAxCCR0

When changing TAxCCR0 while the timer is running and counting in the down direction, the timer continues its descent until it reaches zero. The new period takes effect after the counter counts down to zero.

When the timer is counting in the up direction, and the new period is greater than or equal to the old period or greater than the current count value, the timer counts up to the new period before counting down.

When the timer is counting in the up direction and the new period is less than the current count value, the timer begins counting down. However, one additional count may occur before the counter begins counting down.

### 17.2.3.5 Use of Up/Down Mode

The up/down mode supports applications that require dead times between output signals (see section *Timer\_A Output Unit*). For example, to avoid overload conditions, two outputs driving an H-bridge must never be in a high state simultaneously. In the example shown in Figure 17-9, the  $t_{\text{dead}}$  is:

$$t_{\text{dead}} = t_{\text{timer}} \times (\text{TAxCCR1} - \text{TAxCCR2})$$

Where:

$t_{\text{dead}}$  = Time during which both outputs need to be inactive

$t_{\text{timer}}$  = Cycle time of the timer clock

TAxCCRn = Content of capture/compare register n

The TAxCCRn registers are not buffered. They update immediately when written to. Therefore, any required dead time is not maintained automatically.

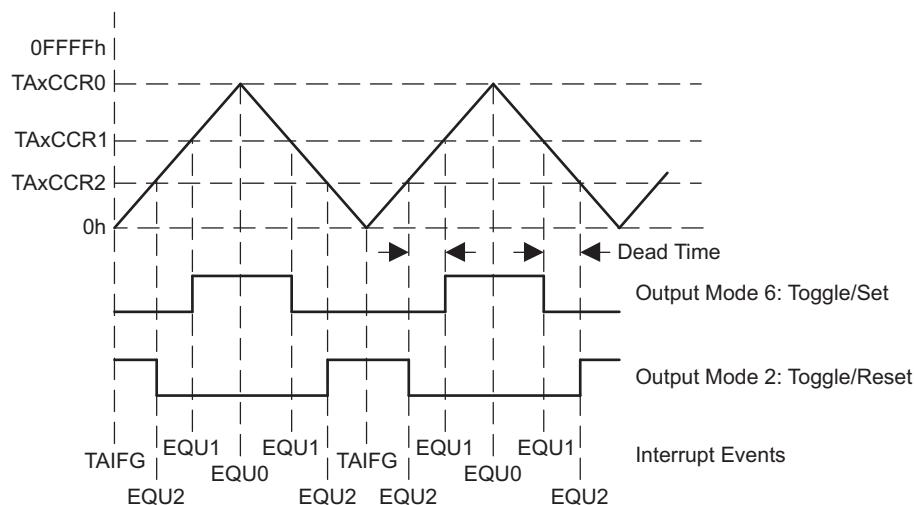


Figure 17-9. Output Unit in Up/Down Mode

### 17.2.4 Capture/Compare Blocks

Up to seven identical capture/compare blocks, TAxCCRn (where n = 0 to 7), are present in Timer\_A. Any of the blocks may be used to capture the timer data or to generate time intervals.

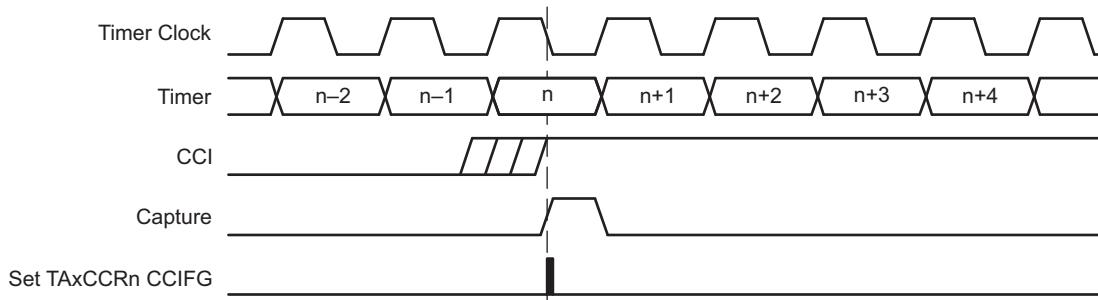
#### 17.2.4.1 Capture Mode

The capture mode is selected when CAP = 1. Capture mode is used to record time events. It can be used for speed computations or time measurements. The capture inputs CCIA and CCIB are connected to external pins or internal signals and are selected with the CCIS bits. The CM bits select the capture edge of the input signal as rising, falling, or both. A capture occurs on the selected edge of the input signal. If a capture occurs:

- The timer value is copied into the TAxCCRn register.
- The interrupt flag CCIFG is set.

The input signal level can be read at any time from the CCI bit. Devices may have different signals connected to CCIA and CCIB. See the device-specific data sheet for the connections of these signals.

The capture signal can be asynchronous to the timer clock and cause a race condition. Setting the SCS bit synchronizes the capture with the next timer clock. Setting the SCS bit to synchronize the capture signal with the timer clock is recommended (see Figure 17-10).

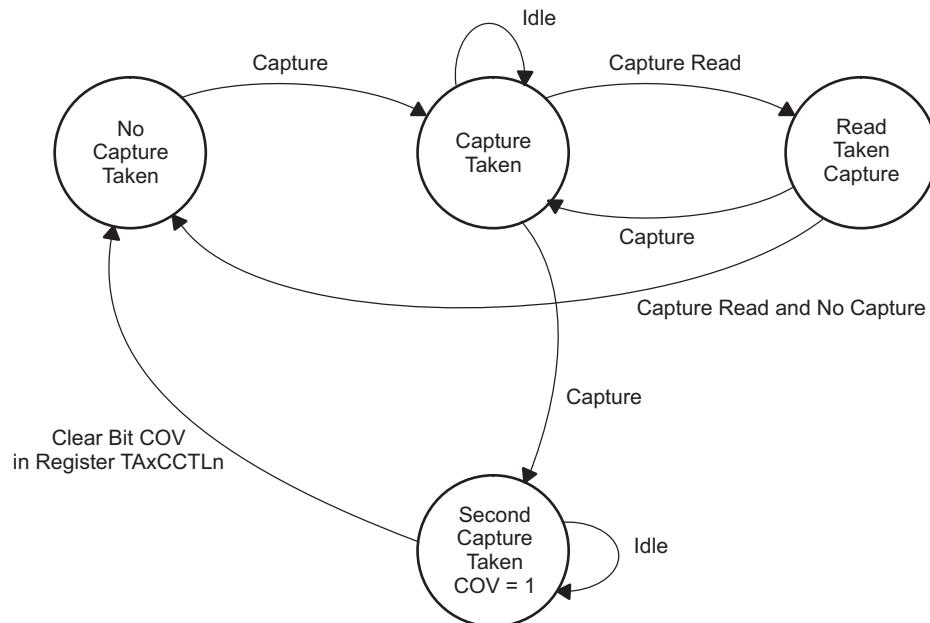


**Figure 17-10. Capture Signal (SCS = 1)**

**NOTE: Changing Capture Inputs**

Changing capture inputs while in capture mode may cause unintended capture events. To avoid this scenario, capture inputs should only be changed when capture mode is disabled (CM = {0} or CAP = 0).

Overflow logic is provided in each capture/compare register to indicate if a second capture was performed before the value from the first capture was read. Bit COV is set when this occurs as shown in [Figure 17-11](#). COV must be reset with software.



**Figure 17-11. Capture Cycle**

#### 17.2.4.1.1 Capture Initiated by Software

Captures can be initiated by software. The CMx bits can be set for capture on both edges. Software then sets CCIS1 = 1 and toggles bit CCIS0 to switch the capture signal between V<sub>CC</sub> and GND, initiating a capture each time CCIS0 changes state:

---

**NOTE: Capture Initiated by Software**

In general, changing capture inputs while in capture mode may cause unintended capture events. For this scenario, switching the capture input between VCC and GND, disabling the capture mode is not required.

---

#### 17.2.4.2 Compare Mode

The compare mode is selected when CAP = 0. The compare mode is used to generate PWM output signals or interrupts at specific time intervals. When TAxR *counts* to the value in a TAxCRRn, where n represents the specific capture/compare register.

- Interrupt flag CCIFG is set.
- Internal signal EQUn = 1.
- EQUn affects the output according to the output mode.
- The input signal CCI is latched into SCCI.

#### 17.2.5 Output Unit

Each capture/compare block contains an output unit. The output unit is used to generate output signals, such as PWM signals. Each output unit has eight operating modes that generate signals based on the EQU0 and EQUn signals.

##### 17.2.5.1 Output Modes

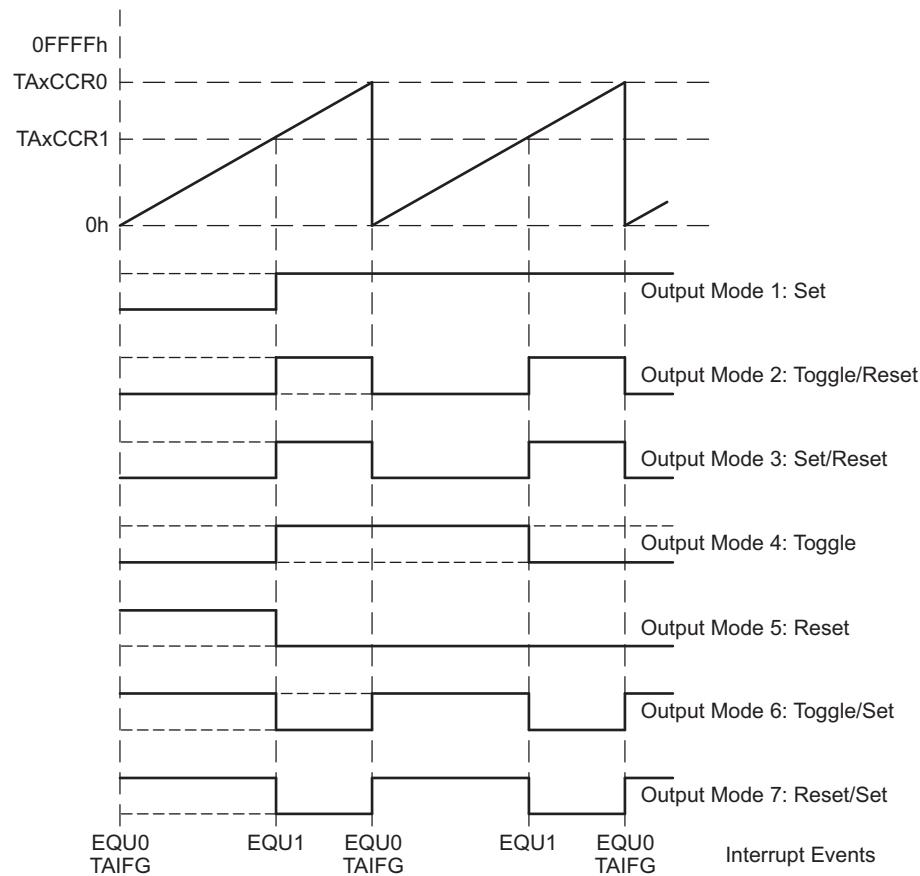
The output modes are defined by the OUTMOD bits (see [Table 17-2](#)). The OUTn signal is changed with the rising edge of the timer clock for all modes except mode 0. Output modes 2, 3, 6, and 7 are not useful for output unit 0 because EQUn = EQU0.

**Table 17-2. Output Modes**

OUTMODx	Mode	Description
000	Output	The output signal OUTn is defined by the OUT bit. The OUTn signal updates immediately when OUT is updated.
001	Set	The output is set when the timer <i>counts</i> to the TAxCRRn value. It remains set until a reset of the timer, or until another output mode is selected and affects the output.
010	Toggle/Reset	The output is toggled when the timer <i>counts</i> to the TAxCRRn value. It is reset when the timer <i>counts</i> to the TAxCCR0 value.
011	Set/Reset	The output is set when the timer <i>counts</i> to the TAxCRRn value. It is reset when the timer <i>counts</i> to the TAxCCR0 value.
100	Toggle	The output is toggled when the timer <i>counts</i> to the TAxCRRn value. The output period is double the timer period.
101	Reset	The output is reset when the timer <i>counts</i> to the TAxCRRn value. It remains reset until another output mode is selected and affects the output.
110	Toggle/Set	The output is toggled when the timer <i>counts</i> to the TAxCRRn value. It is set when the timer <i>counts</i> to the TAxCCR0 value.
111	Reset/Set	The output is reset when the timer <i>counts</i> to the TAxCRRn value. It is set when the timer <i>counts</i> to the TAxCCR0 value.

### 17.2.5.1.1 Output Example—Timer in Up Mode

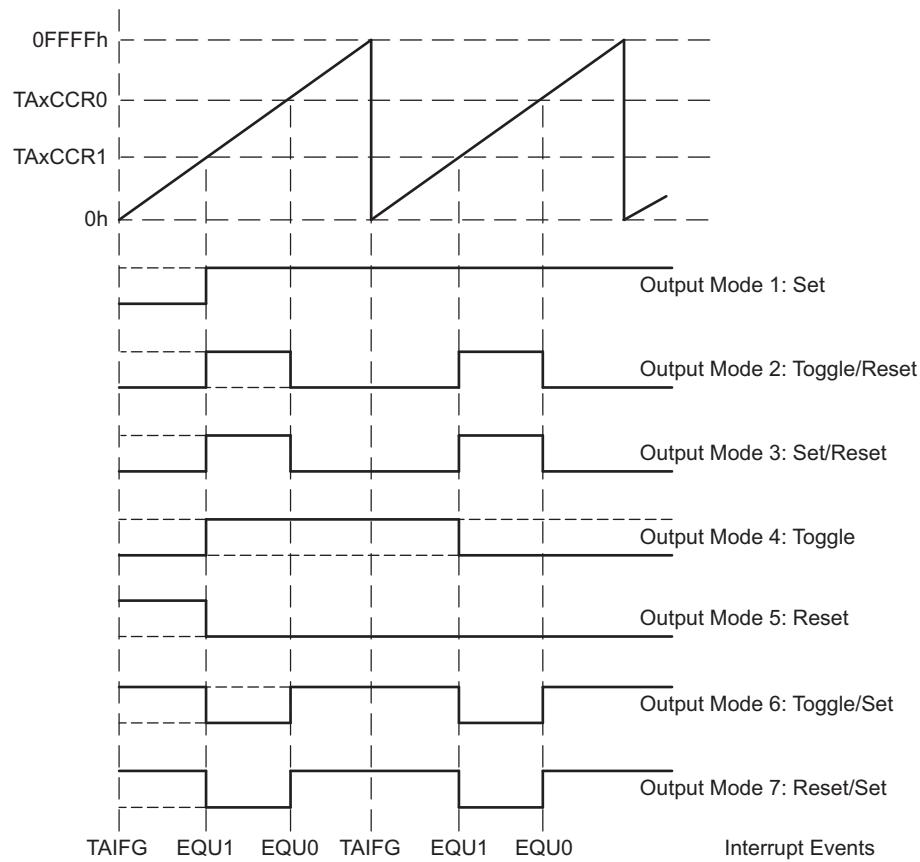
The OUTn signal is changed when the timer *counts* up to the TAxCCRn value and rolls from TAxCCR0 to zero, depending on the output mode. Figure 17-12 shows an example using TAxCCR0 and TAxCCR1.



**Figure 17-12. Output Example – Timer in Up Mode**

### 17.2.5.1.2 Output Example – Timer in Continuous Mode

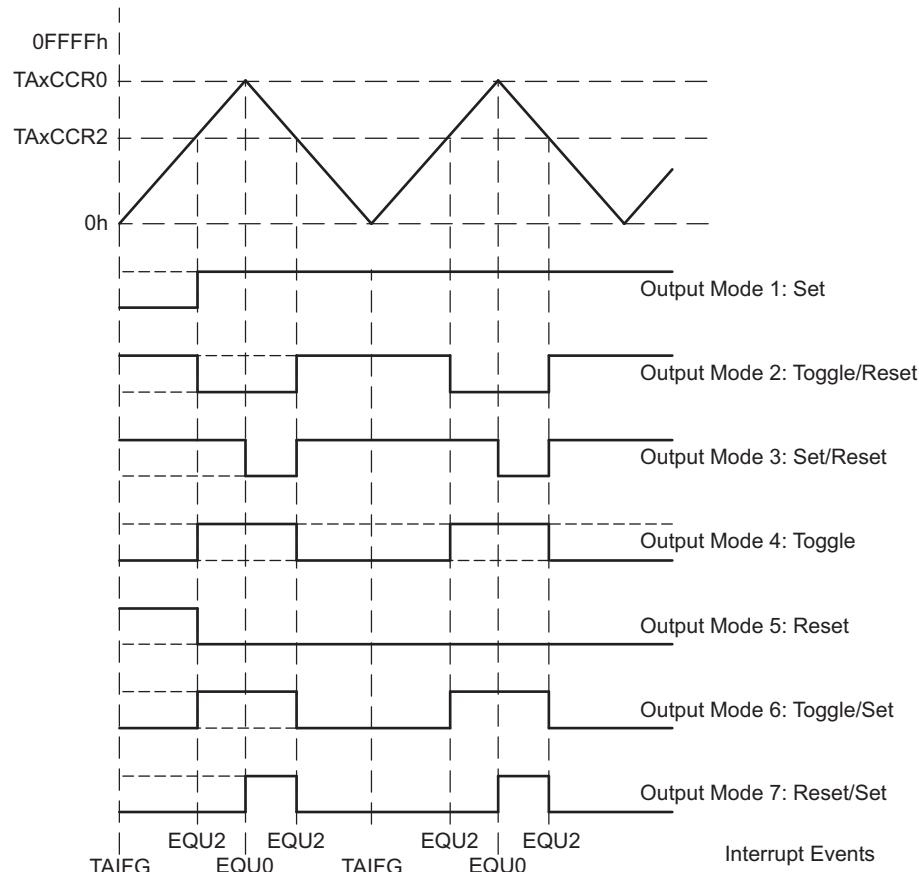
The OUTn signal is changed when the timer reaches the TAxCCR0 and TAxCCR1 values, depending on the output mode. Figure 17-13 shows an example using TAxCCR0 and TAxCCR1.



**Figure 17-13. Output Example – Timer in Continuous Mode**

### 17.2.5.1.3 Output Example – Timer in Up/Down Mode

The OUTn signal changes when the timer equals TAxCCRn in either count direction and when the timer equals TAxCCR0, depending on the output mode. [Figure 17-14](#) shows an example using TAxCCR0 and TAxCCR2.



**Figure 17-14. Output Example – Timer in Up/Down Mode**

---

**NOTE: Switching between output modes**

When switching between output modes, one of the OUTMOD bits should remain set during the transition, unless switching to mode 0. Otherwise, output glitching can occur, because a NOR gate decodes output mode 0. A safe method for switching between output modes is to use output mode 7 as a transition state:

---

## 17.2.6 Timer\_A Interrupts

Two interrupt vectors are associated with the 16-bit Timer\_A module:

- TAxCCR0 interrupt vector for TAxCCR0 CCIFG
- TAxIV interrupt vector for all other CCIFG flags and TAIFG

In capture mode, any CCIFG flag is set when a timer value is captured in the associated TAxCCRn register. In compare mode, any CCIFG flag is set if TAxR *counts* to the associated TAxCCRn value. Software may also set or clear any CCIFG flag. All CCIFG flags request an interrupt when their corresponding CCIE bit is set.

### 17.2.6.1 TAxIV, Interrupt Vector Generator

The TAxCCRRy CCIFG flags and TAIFG flags are prioritized and combined to source a single interrupt vector. The interrupt vector register TAxIV is used to determine which flag requested an interrupt.

The highest-priority enabled interrupt generates a number in the TAxIV register (see register description). This number can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled Timer\_A interrupts do not affect the TAxIV value.

Any access, read or write, of the TAxIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. For example, if the TAxCCR1 and TAxCCR2 CCIFG flags are set when the interrupt service routine accesses the TAxIV register, TAxCCR1 CCIFG is reset automatically. After the completion of TAxCCR1 CCIFG interrupt service routine, the TAxCCR2 CCIFG flag generates another interrupt.

## 17.3 Timer\_A Registers

Timer\_A registers are listed in [Table 17-3](#) for the largest configuration available. The base address can be found in the device-specific data sheet.

**Table 17-3. Timer\_A Registers**

Offset	Acronym	Register Name	Section
00h	TAxCTL	Timer_Ax Control	<a href="#">Section 17.3.1</a>
02h to 0Eh	TAxCCTL0 to TAxCCTL6	Timer_Ax Capture/Compare Control 0 to Timer_Ax Capture/Compare Control 6	<a href="#">Section 17.3.3</a>
10h	TAxR	Timer_Ax Counter	<a href="#">Section 17.3.2</a>
12h to 1Eh	TAxCCR0 to TAxCCR6	Timer_Ax Capture/Compare 0 to Timer_Ax Capture/Compare 6	<a href="#">Section 17.3.4</a>
2Eh	TAxIV	Timer_Ax Interrupt Vector	<a href="#">Section 17.3.5</a>
20h	TAxEX0	Timer_Ax Expansion 0	<a href="#">Section 17.3.6</a>

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through half-word (16 bit) access.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 17.3.1 TAxCTL Register

Timer\_Ax Control Register

**Figure 17-15. TAxCTL Register**

15	14	13	12	11	10	9	8
Reserved						TASSEL	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7						1	0
ID	MC		Reserved		TACLR	TAIE	TAIFG
rw-0	rw-0	rw-0	rw-0	rw-0	w-0	rw-0	rw-0

**Table 17-4. TAxCTL Register Description**

Bit	Field	Type	Reset	Description
15-10	Reserved	RW	0h	Reserved
9-8	TASSEL	RW	0h	Timer_A clock source select 00b = TAxCLK 01b = ACLK 10b = SMCLK 11b = INCLK
7-6	ID	RW	0h	Input divider. These bits along with the TAIDEX bits select the divider for the input clock. 00b = /1 01b = /2 10b = /4 11b = /8
5-4	MC	RW	0h	Mode control. Setting MCx = 00h when Timer_A is not in use conserves power. 00b = Stop mode: Timer is halted 01b = Up mode: Timer counts up to TAxCCR0 10b = Continuous mode: Timer counts up to 0FFFFh 11b = Up/down mode: Timer counts up to TAxCCR0 then down to 0000h
3	Reserved	RW	0h	Reserved
2	TACLR	RW	0h	Timer_A clear. Setting this bit resets TAxR, the timer clock divider logic, and the count direction. The TACLR bit is automatically reset and is always read as zero.
1	TAIE	RW	0h	Timer_A interrupt enable. This bit enables the TAIFG interrupt request. 0b = Interrupt disabled 1b = Interrupt enabled
0	TAIFG	RW	0h	Timer_A interrupt flag 0b = No interrupt pending 1b = Interrupt pending

### **17.3.2 TAxR Register**

Timer\_A Counter Register

**Figure 17-16. TAxR Register**

15	14	13	12	11	10	9	8
TAxR							
rw-0							
TAxR							
rw-0							

**Table 17-5. TAxR Register Description**

Bit	Field	Type	Reset	Description
15-0	TAxR	RW	0h	Timer_A register. The TAxR register is the count of Timer_A.

### 17.3.3 TAxCCTL0 to TAxCCTL6 Register

Timer\_Ax Capture/Compare Control 0 Register to Timer\_Ax Capture/Compare Control 6

**Figure 17-17. TAxCCTL0 to TAxCCTL6 Register**

15	14	13	12	11	10	9	8
CM		CCIS		SCS	SCCI	Reserved	CAP
rw-0	rw-0	rw-0	rw-0	rw-0	r-0	r-0	rw-0
7	6	5	4	3	2	1	0
	OUTMOD		CCIE	CCI	OUT	COV	CCIFG
rw-0	rw-0	rw-0	rw-0	r	rw-0	rw-0	rw-0

**Table 17-6. TAxCCTL0 to TAxCCTL6 Register Description**

Bit	Field	Type	Reset	Description
15-14	CM	RW	0h	Capture mode 00b = No capture 01b = Capture on rising edge 10b = Capture on falling edge 11b = Capture on both rising and falling edges
13-12	CCIS	RW	0h	Capture/compare input select. These bits select the TAxCCTL0 input signal. See the device-specific data sheet for specific signal connections. 00b = CCIA 01b = CCIB 10b = GND 11b = VCC
11	SCS	RW	0h	Synchronize capture source. This bit is used to synchronize the capture input signal with the timer clock. 0b = Asynchronous capture 1b = Synchronous capture
10	SCCI	RW	0h	Synchronized capture/compare input. The selected CCI input signal is latched with the EQUx signal and can be read via this bit.
9	Reserved	R	0h	Reserved. Reads as 0.
8	CAP	RW	0h	Capture mode 0b = Compare mode 1b = Capture mode
7-5	OUTMOD	RW	0h	Output mode. Modes 2, 3, 6, and 7 are not useful for TAxCCTL0 because EQU0 = EQU0. 000b = OUT bit value 001b = Set 010b = Toggle/reset 011b = Set/reset 100b = Toggle 101b = Reset 110b = Toggle/set 111b = Reset/set
4	CCIE	RW	0h	Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag. 0b = Interrupt disabled 1b = Interrupt enabled
3	CCI	R	0h	Capture/compare input. The selected input signal can be read by this bit.
2	OUT	RW	0h	Output. For output mode 0, this bit directly controls the state of the output. 0b = Output low 1b = Output high

**Table 17-6. TAxCCTL0 to TAxCCTL6 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
1	COV	RW	0h	Capture overflow. This bit indicates a capture overflow occurred. COV must be reset with software. 0b = No capture overflow occurred 1b = Capture overflow occurred
0	CCIFG	RW	0h	Capture/compare interrupt flag 0b = No interrupt pending 1b = Interrupt pending

### 17.3.4 TAxCRR0 to TAxCRR6 Register

Timer\_Ax Capture/Compare 0 Register to Timer\_Ax Capture/Compare 6 Register

**Figure 17-18. TAxCRR0 to TAxCRR6 Register**

15	14	13	12	11	10	9	8
TAxCRRn							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
TAxCRRn							
7	6	5	4	3	2	1	0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 17-7. TAxCRR0 to TAxCRR6 Register Description**

Bit	Field	Type	Reset	Description
15-0	TAxCRR0	RW	0h	Compare mode: TAxCRRn holds the data for the comparison to the timer value in the Timer_A Register, TAxR. Capture mode: The Timer_A Register, TAxR, is copied into the TAxCRRn register when a capture is performed.

### 17.3.5 TAIV Register

Timer\_Ax Interrupt Vector Register

**Figure 17-19. TAIV Register**

15	14	13	12	11	10	9	8
TAIV							
r0	r0	r0	r0	r0	r0	r0	r0
TAIV							
r0	r0	r0	r0	r-0	r-0	r-0	r0

**Table 17-8. TAIV Register Description**

Bit	Field	Type	Reset	Description
15-0	TAIV	R	0h	Timer_A interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Capture/compare 1; Interrupt Flag: TAxCCR1 CCIFG; Interrupt Priority: Highest 04h = Interrupt Source: Capture/compare 2; Interrupt Flag: TAxCCR2 CCIFG 06h = Interrupt Source: Capture/compare 3; Interrupt Flag: TAxCCR3 CCIFG 08h = Interrupt Source: Capture/compare 4; Interrupt Flag: TAxCCR4 CCIFG 0Ah = Interrupt Source: Capture/compare 5; Interrupt Flag: TAxCCR5 CCIFG 0Ch = Interrupt Source: Capture/compare 6; Interrupt Flag: TAxCCR6 CCIFG 0Eh = Interrupt Source: Timer overflow; Interrupt Flag: TAxCTL TAIFG; Interrupt Priority: Lowest

### 17.3.6 TAxE0 Register

Timer\_Ax Expansion 0 Register

**Figure 17-20. TAxE0 Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						TAIDEX <sup>(1)</sup>	
r0	r0	r0	r0	r0	rw-0	rw-0	rw-0

<sup>(1)</sup> After programming TAIDEX bits and configuration of the timer, set TACLR bit to ensure proper reset of the timer divider logic.

**Table 17-9. TAxE0 Register Description**

Bit	Field	Type	Reset	Description
15-3	Reserved	R	0h	Reserved. Reads as 0.
2-0	TAIDEX	RW	0h	Input divider expansion. These bits along with the ID bits select the divider for the input clock. 000b = Divide by 1 001b = Divide by 2 010b = Divide by 3 011b = Divide by 4 100b = Divide by 5 101b = Divide by 6 110b = Divide by 7 111b = Divide by 8

## **Real-Time Clock (RTC\_C)**

The Real-Time Clock (RTC\_C) module provides clock counters with calendar mode, a flexible programmable alarm, offset calibration, and a provision for temperature compensation. The RTC\_C also supports operation in Low-Power modes like LPM3 and LPM3.5. This chapter describes the RTC\_C module.

Topic	Page
18.1 RTC_C Introduction .....	625
18.2 RTC_C Operation.....	627
18.3 RTC_C Registers .....	634

## 18.1 RTC\_C Introduction

The RTC\_C module provides configurable clock counters.

RTC\_C features include:

- Real-time clock and calendar mode providing seconds, minutes, hours, day of week, day of month, month, and year (including leap year correction)
- Protection for real-time clock registers
- Interrupt capability
- Selectable BCD or binary format
- Programmable alarms
- Real-time clock calibration for crystal offset error
- Real-time clock compensation for crystal temperature drift
- Operation in Low-Power modes: LPM3 and LPM3.5.

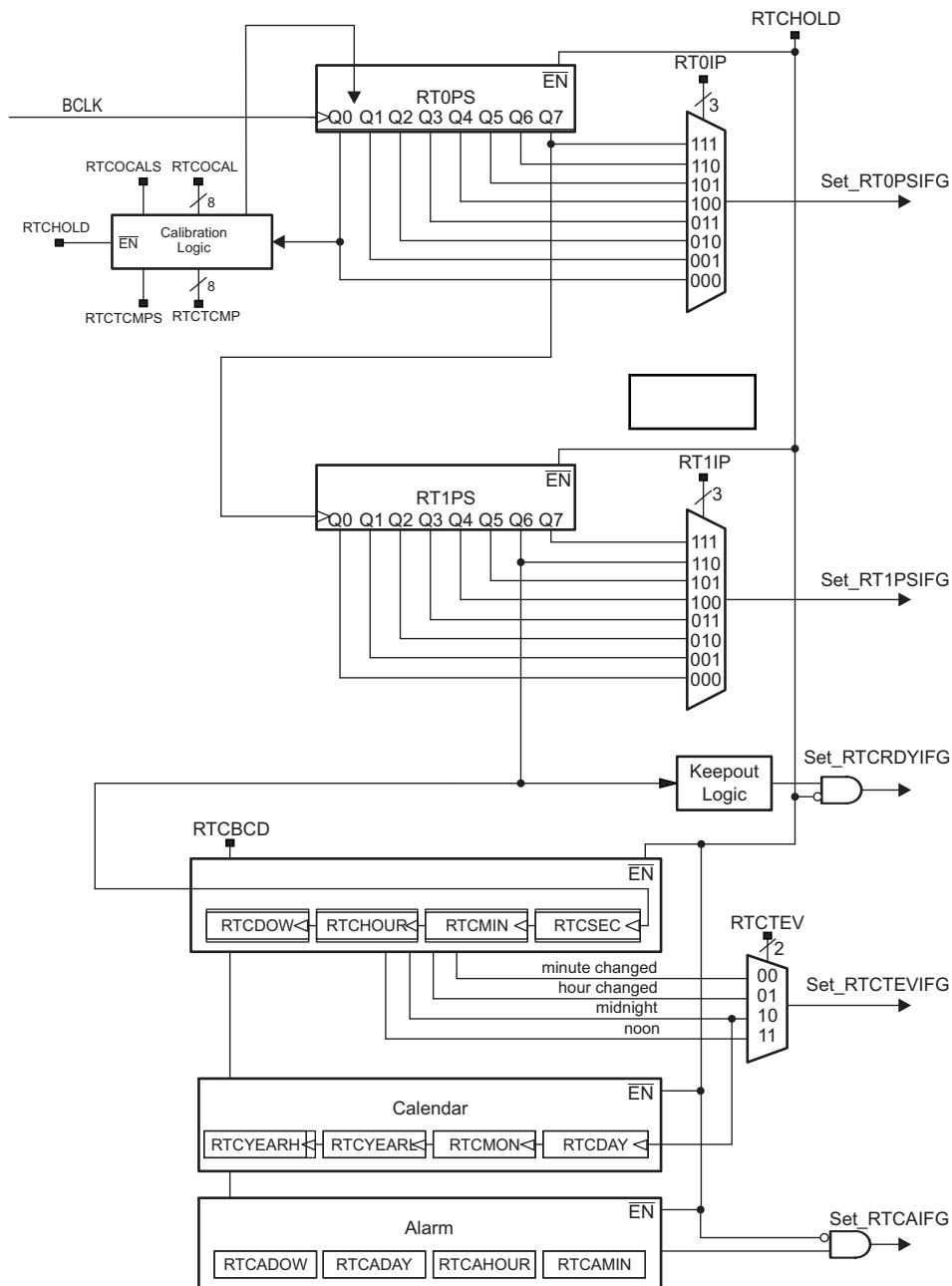
---

**NOTE: Real-time clock initialization**

Most RTC\_C module registers have no initial condition. These registers must be configured by user software before use.

---

Figure 18-1 shows the RTC\_C block diagram.



**Figure 18-1. RTC\_C Block Diagram**

## 18.2 RTC\_C Operation

### 18.2.1 Calendar Mode

The RTC\_C module provides seconds, minutes, hours, day of week, day of month, month, and year in either BCD or hexadecimal format. The calendar includes a leap-year algorithm that considers all years evenly divisible by four as leap years. This algorithm is accurate from the year 1901 through 2099. The RTC registers must be configured by user software before use.

### 18.2.2 Real-Time Clock and Prescale Dividers

The prescale dividers, RT0PS and RT1PS, are automatically configured to provide a 1-second clock interval for the RTC. The RTC clock source (BCLK) must be operated at 32768 Hz, nominal for proper RTC operation. RT0PS is sourced from the backup domain clock (BCLK). The output of RT0PS /256 (Q7) is used to source RT1PS. RT1PS is further divider and the /128 output sources the real-time clock counter registers providing the required 1-second time interval.

When RTCBCD = 1, BCD format is selected for the calendar registers. It is possible to switch between BCD and hexadecimal format while the RTC is counting.

Setting RTCHOLD halts the real-time counters and prescale counters, RT0PS and RT1PS.

---

**NOTE: For reliable update to all Calendar Mode registers**

Keep RTCHOLD = '1' before writing into ANY of the calendar/prescalar registers (RTCP0/1, RTCSEC, RTCMIN, RTCHOUR, RTCDAY, RTCADOW, RTCMON, RTCYEAR).

---

### 18.2.3 Real-Time Clock Alarm Function

The RTC\_C module provides for a flexible alarm system. There is a single user-programmable alarm that can be programmed based on the settings contained in the alarm registers for minutes, hours, day of week, and day of month.

Each alarm register contains an alarm enable (AE) bit that can be used to enable the respective alarm register. By setting AE bits of the various alarm registers, a variety of alarm events can be generated.

- Example 1: A user wishes to set an alarm every hour at 15 minutes past the hour, i.e., 00:15:00, 01:15:00, 02:15:00, etc. This is possible by setting RTCAMIN to 15. By setting the AE bit of the RTCAMIN and clearing all other AE bits of the alarm registers, the alarm is enabled. When enabled, the RTCAIFG is set when the count transitions from 00:14:59 to 00:15:00, 01:14:59 to 01:15:00, 02:14:59 to 02:15:00, etc.
- Example 2: A user wishes to set an alarm every day at 04:00:00. This is possible by setting RTCAHOUR to 4. By setting the AE bit of the RTCHOUR and clearing all other AE bits of the alarm registers, the alarm is enabled. When enabled, the RTCAIFG is set when the count transitions from 03:59:59 to 04:00:00.
- Example 3: A user wishes to set an alarm for 06:30:00. RTCAHOUR would be set to 6 and RTCAMIN would be set to 30. By setting the AE bits of RTCAHOUR and RTCAMIN, the alarm is enabled. Once enabled, the RTCAIFG is set when the time count transitions from 06:29:59 to 06:30:00. In this case, the alarm event occurs every day at 06:30:00.
- Example 4: A user wishes to set an alarm every Tuesday at 06:30:00. RTCADOW would be set to 2, RTCAHOUR would be set to 6 and RTCAMIN would be set to 30. By setting the AE bits of RTCADOW, RTCAHOUR and RTCAMIN, the alarm is enabled. Once enabled, the RTCAIFG is set when the time count transitions from 06:29:59 to 06:30:00 and the RTCADOW transitions from 1 to 2.
- Example 5: A user wishes to set an alarm the fifth day of each month at 06:30:00. RTCDAY would be set to 5, RTCAHOUR would be set to 6 and RTCAMIN would be set to 30. By setting the AE bits of RTCDAY, RTCAHOUR and RTCAMIN, the alarm is enabled. Once enabled, the RTCAIFG is set when the time count transitions from 06:29:59 to 06:30:00 and the RTCDAY equals 5.

**NOTE: Invalid alarm settings**

Invalid alarm settings are not checked by hardware. It is the user's responsibility that valid alarm settings are entered.

**NOTE: Invalid time and date values**

Writing of invalid date and/or time information or data values outside the legal ranges specified in the RTCSEC, RTCMIN, RTC HOUR, RTCDAY, RTCDOW, RTCYEARH, RTCYEARL, RTCAMIN, RTCAHOUR, RTCADAY, and RTCADOW registers can result in unpredictable behavior.

**NOTE: Setting the alarm**

Before setting an initial alarm, all alarm registers including the AE bits should be cleared.

To prevent potential erroneous alarm conditions from occurring, the alarms should be disabled by clearing the RTCAIE, RTCAIFG, and AE bits before writing initial or new time values to the RTC time registers.

### 18.2.4 Real-Time Clock Protection

The RTC registers are key protected to ensure clock integrity and module configuration against software crash or from runaway code. Key protection does not apply for read from RTC registers. That is, any RTC register can be read at any time without having to unlock the module. But some predefined registers of RTC are key protected for write access. The control registers, clock registers, calendar register, prescale timer registers, and offset error calibration registers are protected. RTC alarm function registers, prescale timer control registers, interrupt vector register, and temperature compensation registers are not protected. RTC registers that are not protected can be written at any time without unlocking the module. See tables in [Section 18.3](#) for details on registers covered under the protection scheme.

RTCCTL0\_H register implements key protection and controls lock or unlock state of the module. When this register is written with correct key, 0A5h, the module is unlocked and unlimited write access possible to the RTC registers. Once the module is unlocked, it remains unlocked until user writes any incorrect key or until the module is reset. A read from RTCCTL0\_H register returns value 96h. Write access to any protected registers of RTC is ignored when the module is locked.

### 18.2.5 Reading or Writing Real-Time Clock Registers

Because the system clock may in fact be asynchronous to the RTC clock source, special care must be used when accessing the real-time clock registers.

The real-time clock registers are updated once per second. To prevent reading any real-time clock register at the time of an update that could result in an invalid time being read, a keep-out window is provided. The keep-out window is centered approximately 128/32768 seconds around the update transition. The read only RTCRDY bit is reset during the keep-out window period and set outside the keep-out the window period. Any read of the clock registers while RTCRDY is reset is considered to be potentially invalid, and the time read should be ignored.

An easy way to safely read the real-time clock registers is to utilize the RTCRDYIFG interrupt flag. Setting RTCRDYIE enables the RTCRDYIFG interrupt. Once enabled, an interrupt is generated based on the rising edge of the RTCRDY bit, causing the RTCRDYIFG to be set. At this point, the application has nearly a complete second to safely read any or all of the real-time clock registers. This synchronization process prevents reading the time value during transition. The RTCRDYIFG flag is reset automatically when the interrupt is serviced or can be reset with software.

**NOTE: Reading or writing real-time clock registers**

When the counter clock is asynchronous to the CPU clock, any read from any RTCSEC, RTCMIN, RTC HOUR, RTCDOW, RTC DAY, RTC MON, RTC YEARL, or RTC YEARH register while the RTCRDY is reset may result in invalid data being read. To safely read the counting registers, either polling of the RTCRDY bit or the synchronization procedure previously described can be used. Alternatively, the counter register can be read multiple times while operating, and a majority vote taken in software to determine the correct reading. Reading the RT0PS and RT1PS can only be handled by reading the registers multiple times and a majority vote taken in software to determine the correct reading.

Any write to any counting register takes effect immediately. However, the clock is stopped during the write. In addition, RT0PS and RT1PS registers are reset. This could result in losing up to 1 second during a write. Writing of data outside the legal ranges or invalid time stamp combinations results in unpredictable behavior.

### 18.2.6 Real-Time Clock Interrupts

At least six sources for interrupts are available, namely RT0PSIFG, RT1PSIFG, RTCRDYIFG, RTCTEVIFG, RTCAIFG, and RTCOFIFG. These flags are prioritized and combined to source a single interrupt vector. The interrupt vector register (RTCIV) is used to determine which flag requested an interrupt.

The highest-priority enabled interrupt generates a number in the RTCIV register (see register description). This number can be evaluated or added to the program counter (PC) to automatically enter the appropriate software routine. Disabled RTC interrupts do not affect the RTCIV value.

Writes into RTCIV register clear all pending interrupt conditions. Reads from RTCIV register clear the highest priority pending interrupt condition. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. In addition, all flags can be cleared by software.

The user-programmable alarm event sources the real-time clock interrupt, RTCAIFG. Setting RTCAIE enables the interrupt. In addition to the user-programmable alarm, the RTC\_C module provides for an interval alarm that sources real-time clock interrupt, RTCTEVIFG. The interval alarm can be selected to cause an alarm event when RTCMIN changed or RTC HOUR changed, every day at midnight (00:00:00) or every day at noon (12:00:00). The event is selectable with the RTCTEV bits. Setting the RTCTEVIE bit enables the interrupt.

The RTCRDY bit sources the real-time clock interrupt, RTCRDYIFG, and is useful in synchronizing the read of time registers with the system clock. Setting the RTCRDYIE bit enables the interrupt.

RT0PSIFG can be used to generate interrupt intervals selectable by the RT0IP bits. RT0PS is sourced with BCLK at 32768 Hz, so intervals of 16384 Hz, 8192 Hz, 4096 Hz, 2048 Hz, 1024 Hz, 512 Hz, 256 Hz, or 128 Hz are possible. Setting the RT0PSIE bit enables the interrupt.

RT1PSIFG can be used to generate interrupt intervals selectable by the RT1IP bits. RT1PS is sourced with the output of RT0PS, which is 128 Hz (32768/256 Hz). Therefore, intervals of 64 Hz, 32 Hz, 16 Hz, 8 Hz, 4 Hz, 2 Hz, 1 Hz, or 0.5 Hz are possible. Setting the RT1PSIE bit enables the interrupt.

**NOTE: Changing RT0IP or RT1IP**

Changing the settings of the interrupt interval bits RT0IP or RT1IP while the corresponding prescaler is running or is stopped in a non-zero state can result in setting the corresponding interrupt flags.

The RTCOFIFG bit flags the failure of the 32-kHz crystal oscillator connected to the BCLK line if a fault happens to this oscillator while the design is in the Low-Power modes. When the oscillator faults out, the failsafe is activated and the failsafe clock for BCLK is supplied to the RTC. This failsafe mechanism is effective in both active and low-power modes. Refer to the *Clock System (CS)* chapter for the details. The main purpose of the RTCOFIFG flag is to wake up the CPU from low-power mode of operation, because the fault bit corresponding to the 32-kHz oscillator is not available for CPU interrupt in low-power modes if an oscillator failure occurs.

### 18.2.7 Real-Time Clock Calibration for Crystal Offset Error

The RTC\_C module can be calibrated for crystal manufacturing tolerance or offset error for higher time keeping accuracy. The crystal frequency error of up to  $\pm 240\text{ppm}$  can be calibrated smoothly over a period of 60 seconds. RTCOCAL\_L register is used to adjust the frequency. The calibration value is written into RTCOCAL\_L register and each LSB in this register represent approximately  $\pm 1\text{ppm}$  correction based on RTCOCALS bit in RTCOCAL\_H register. When RTCOCALS bit is set (up calibration), each LSB in RTCOCAL\_L represent  $+1\text{ppm}$  adjustment and when RTCOCALS is cleared (down calibration), each LSB in RTCOCAL\_L represent  $-1\text{ppm}$  adjustment to frequency. Both RTCOCAL\_L and RTCOCAL\_H registers are protected and requires RTC to be unlocked before writing into these registers.

#### 18.2.7.1 Calibration Frequency

To calibrate the frequency, the RTCCLK output signal is available at a pin. RTCCALFx bits in RTCCTL3 register can be used to select the frequency rate of the output signal. When RTCCALFx = 00, no signal is output on RTCCLK pin. The other settings of RTCCALFx select one of the three frequencies 512Hz, 256Hz or 1Hz. RTCCLK can be measured and the result of this measurement can be applied to the RTCOCALS and RTCOCALx bits to effectively reduce the initial offset of the clock.

##### 18.2.7.1.1 Calibration Mechanism

RTCOCAL\_L is an 8 bit register. Software can write up to a value of 256ppm into this register but the maximum frequency error that can be corrected is only 240ppm. Software should take care of writing legal values into this register. Read from RTCOCAL always returns the value that was written by software. Real-time clock offset error calibration is inactive when RTC is not enabled (RTCHOLD = 0) or when RTCOCALx bits are zero. RTCOCAL should only be written when RTCHOLD=1. Writing RTCOCAL resets temperature compensation to zero.

In RTC, the offset error calibration takes place over a period of 60 seconds. To achieve approximately  $\pm 1\text{ppm}$  correction, the 16kHz clock (Q0 output of RT0PS) is adjusted to add or subtract 1 clock pulse. For  $+1\text{ppm}$  correction, 1 clock pulse is added to 16kHz clock and for  $-1\text{ppm}$  correction, 1 clock pulse is subtracted from 16kHz clock. This correction happens once every quarter second until the programmed ppm error is compensated.

$$f_{BCLK,meas} < 32768\text{Hz} \Rightarrow \text{RTCOCALS} = 1, \text{RTCOCALx} = \text{Round}(60 \times 16384 \times (1-f_{BCLK,meas}/32768))$$

$$f_{BCLK,meas} \geq 32768\text{Hz} \Rightarrow \text{RTCOCALS} = 0, \text{RTCOCALx} = \text{Round}(60 \times 16384 \times (1-f_{BCLK,meas}/32768))$$

As an example for up calibration, when the measured frequency is 511.9658Hz against the reference frequency of 512Hz, the frequency error is approximately 67ppm low. In order to increase the frequency by 67ppm, RTCOCALS should be set, and RTCOCALx should be set to Round  $(60 \times 16384 \times (1 - 511.9658/512)) = 66$ .

As an example for down calibration, when the measured frequency is 512.0241Hz against the reference frequency of 512Hz, the frequency error is approximately 47ppm high. In order to decrease the frequency by 47ppm, RTCOCALS should be cleared, and RTCOCALx should be set to Round  $(60 \times 16384 \times (1 - 512.0241/512)) = 46$ .

All three possible output frequencies 512Hz, 256Hz, and 1Hz at RTCCLK pin are affected by calibration settings. RT0PS interrupt triggered by RT0PS-Q0 (RT0IPx = 000) is based on un-calibrated clock while RT0PS interrupt triggered by RT0PS – Q1 to Q7 (RT0IPx ≠ 000) is based on calibrated clock. RT1PS interrupt (RT1PSIFG) and RTC counter interrupt (RTCTEVIFG) are also based on calibrated clock.

### 18.2.8 Real-Time Clock Compensation for Crystal Temperature Drift

The frequency output of the crystal varies considerably due to drift in temperature. It would be necessary to compensate the real-time clock for this temperature drift for higher time keeping accuracy from standard crystals. A hybrid software and hardware approach can be followed to achieve temperature compensation for RTC.

The software can make use of an (on-chip) temperature sensor to measure the temperature at desired intervals (for example, once every few seconds or minutes). The temperature sensor parameters are calibrated at production and the values are stored in the device descriptor (TLV). Using the temperature sensor parameters and the measured temperature, software can do parabolic calculations to find out the corresponding frequency error in ppm.

This frequency error can be written into RTCTCMP\_L register for temperature compensation. RTCTCMP\_L is an 8 bit register that can cover the frequency error up to  $\pm 240$ ppm. Each LSB in this register represent  $\pm 1$ ppm based on RTCTCMPS bit in RTCTCMP\_H register. When RTCTCMPS bit is set, each LSB in RTCTCMP represent +1ppm adjustment (up calibration) and when RTCTCMPS is cleared, each LSB in RTCTCMP represent -1ppm adjustment (down calibration). RTCTCMP register is not protected and can be written any time without having to unlock RTC.

#### 18.2.8.1 Temperature Compensation

RTCTCMP\_L is an 8 bit register. Software can write up to value of 256ppm into this register but the maximum frequency error that can be corrected including the crystal offset error is only 240ppm. Real-time clock temperature compensation is inactive when RTC is not enabled (RTCHOLD = 0) or when RTCTCMPx bits are zero

When the temperature compensation value is written into RTCTCMP\_L, it is added with offset error calibration value and the resulting value is taken into account from next calibration cycle onwards. The ongoing calibration cycle is not affected by writes into RTCTCMP register. The maximum frequency error that can be corrected to account for both offset error and temperature variation is  $\pm 240$ ppm. This means the sign addition of offset error value and temperature compensation value should not exceed maximum of  $\pm 240$ ppm otherwise the excess value above  $\pm 240$ ppm is ignored by hardware. Reading from RTCTCMP register at any time returns the cumulative value which is the signed addition of RTCOCALx and RTCTCMPx values. (Writing RTCOCAL reset the temperature compensation value to zero.)

For example, when RTCOCAL value is +150ppm and the value written into RTCTCMP is +200ppm the effective value taken in for next calibration cycle is +240ppm. Software is expected to do temperature measurement at certain regularity, calculate the frequency error and write into RTCTCMP register to not to exceed the max limit of  $\pm 240$ ppm.

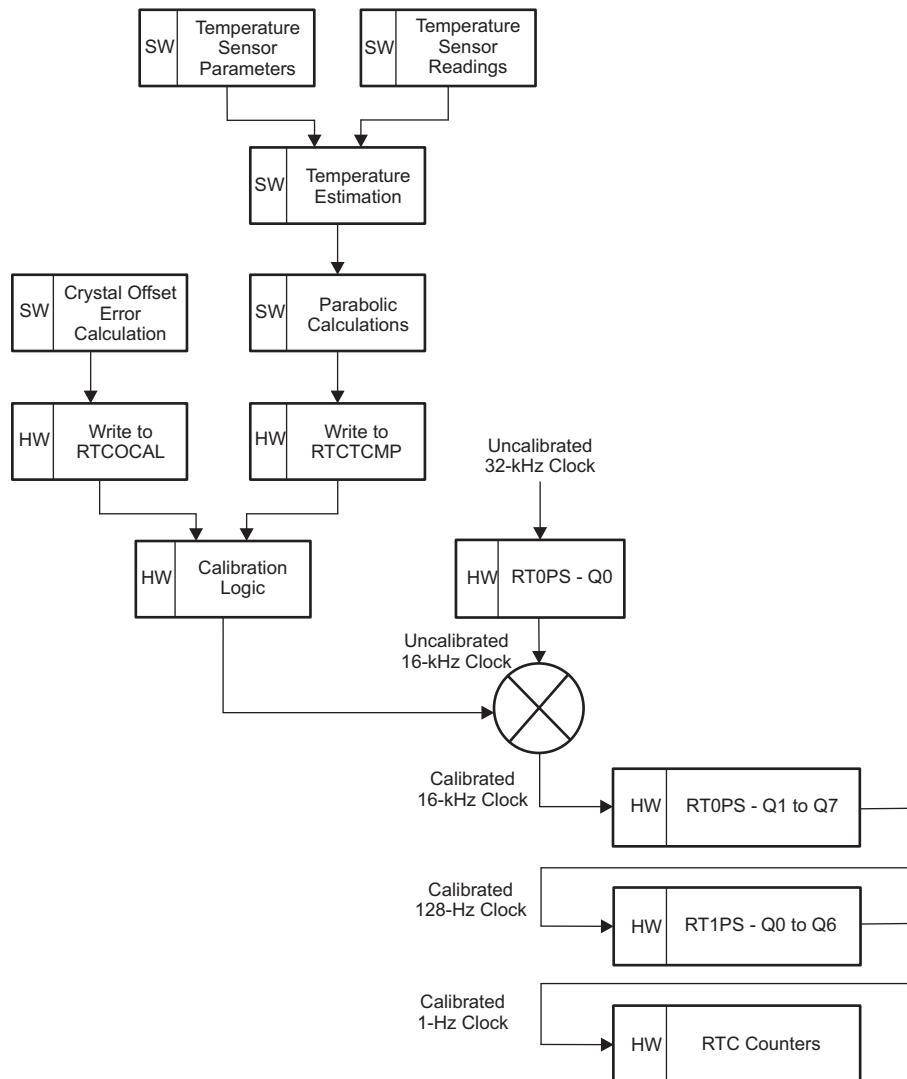
Changing the sign-bit by writing to RTCTCMP\_H is effective only after writing RTCTCMP\_L as well. Thus TI recommends writing the sign-bit together with the compensation value as a 16-bit value into RTCTCMP.

#### 18.2.8.2 Writing to RTCTCMP Register

As the system clock could be asynchronous to the RTC clock source, RTCTCRDY bit in RTCTCMP\_H register should be considered for reliable writing into RTCTCMP register. RTCTCRDY is a read only bit that gets set when the hardware is ready to take in the new temperature compensation value. Write to RTCTCMP should be avoided when RTCTCRDY bit is reset. Writes into RTCTCMP register when RTCTCRDY is reset are ignored.

RTCTCOK is a status bit that indicates if the write to RTCTCMP register is successful or not. RTCTCOK is set if the write to RTCTCMP is successful and reset if the write is unsuccessful. The status remains same until affected by the next write to RTCTCMP register. If the write to RTCTCMP is unsuccessful, then application must write into RTCTCMP again when RTCTCRDY is set.

[Figure 18-2.](#) shows the scheme for real-time clock offset error calibration and temperature compensation.



**Figure 18-2. RTC\_C Offset Error Calibration and Temperature Compensation**

### 18.2.8.3 Temperature Measurement and Updates to RTC

The application may need to perform temperature measurement once every few seconds or once every minute or once in several minutes. Writing to the RTCTCMP register for temperature compensation is effective once per one minute. This means that if the user performs temperature measurement every minute and updates RTCTCMP register with the frequency error, compensation would immediately apply. But if software performs temperature measurement more frequently than once per minute (for example once every 5 seconds), then the application should average the error over one minute and update RTCTCMP register once per minute. If the software performs temperature measurement less frequently than once per minute (for example, once every 5 minutes), then the application must calculate the frequency error for the measured temperature and write into RTCTCMP register. The value written into RTCTCMP in this case would be effective until it is updated again by software.

### 18.2.9 Real-Time Clock Operation in Low-Power Modes

In low-power modes, the power to the switchable domain is off and the power to the backup domain of the device is on. Some of the registers of the RTC\_C module reside in the switchable domain and hence lose their content when the device enters low-power modes. However, the RTC continues to function in the low-power modes regardless of the power loss to these registers. When the CPU wakes from the low-power mode of operation, the application must reconfigure these registers. In LPM3, all of the RTC register values are retained due to state retention. This section describes only the entry into LPM3.5, because most of the RTC configuration registers are lost. In LPM3.5, only the counters and calibration registers are retained.

**Table 18-1** lists the registers that are retained during LPM3.5 mode of operation. Also, the configuration of the interrupt enables is latched so that the configured interrupt conditions can cause a wakeup from LPM3.5 as desired. Interrupt flags that are set before entering LPM3.5 are cleared upon entering LPM3.5. (This can happen only if the corresponding interrupt is not enabled). The interrupt flags RTCTEVIFG, RTCAIFG, RT0PSIFG, RT1PSIFG, and RTCOFIFG can be used as RTC wake-up interrupt sources. Any interrupt event that occurs during LPM3.5 is stored in the corresponding flags, but only enabled interrupts cause a wake-up. After restoring the configuration registers (and clearing LOCKBKUP and LOCKLPM5 bits in PCM), the interrupts can be serviced as usual.

The detailed flow is as follows:

1. Set all I/Os to general-purpose I/Os and configure them as needed. Optionally, configure input interrupt pins for wakeup. Configure RTC interrupts for wakeup (set RTCTEVIE, RTCAIE, RT0PSIE, RT1PSIE, or RTCOFIE. If the alarm interrupt is used as wake-up event also, the alarm registers must be configured as needed).
2. Enter LPM3.5 mode with corresponding entry sequence.
3. LOCKBKUP and LOCKLPM5 bits are automatically set by hardware upon entering LPM3.5, the switchable domain's power is turned off now, and all clocks are disabled except for the BCLK as the RTC is enabled with RTCHOLD = 0.
4. A LPM3.5 wake-up event like an edge on a wake-up input pin or an RTC interrupt event starts the power restoration to the switchable domain of the device. All peripheral registers are set to their default conditions and the device reaches the AM\_LDO\_VCORE0 mode of operation. The I/O pin state and the interrupt configuration for the RTC remain locked using the signals LOCKLPM5 and LOCKBKUP bits, respectively.
5. The device can be configured. The I/O configuration and the RTC interrupt configuration that were not retained during LPM3.5 should be restored to the values before entering LPM3.5. Then the LOCKLPM5 and LOCKBKUP bits can be cleared, which releases the I/O pin conditions and the RTC interrupt configuration. Registers that are retained during LPM3.5 should not be altered before LOCKLPM5 and LOCKBKUP are cleared.
6. NVIC interrupt enable register should be configured for port or RTC module if interrupt servicing is desired.

If the RTC is enabled (RTCHOLD = 0), the BCLK along with its clock source remain active during LPM3.5 mode. Also the fault detection for the oscillator remains functional within RTC. If during LPM3.5 a fault occurs and the RTCOFIE was set before entering LPM3.5, a wake-up event is issued.

## 18.3 RTC\_C Registers

The RTC\_C module registers are shown in [Table 18-1](#). This table also shows which registers are key protected and which are retained during LPM3.5. The registers that are retained during LPM3.5 and given with a reset value are not reset on hard-reset; they are reset with hard-reset only when LOCKBKUP = 0. Registers that are not retained during LPM3.5 must be restored after exit from LPM3.5 before clearing the LOCKBKUP bit in PCM.

The base address for the RTC\_C module registers can be found in the device-specific data sheet. The address offsets are shown in [Table 18-1](#).

---

**NOTE:** Most registers have word or byte register access. For a generic register ANYREG, the suffix "\_L" (ANYREG\_L) refers to the lower byte of the register (bits 0 through 7). The suffix "\_H" (ANYREG\_H) refers to the upper byte of the register (bits 8 through 15).

---

**Table 18-1. RTC\_C Registers**

Offset	Acronym	Register Name	Type	Key Protected	LPM3.5 Retention	Section
00h	RTCCTL0	Real-Time Clock Control 0	Read/write	yes	not retained	<a href="#">Section 18.3.1</a>
00h	RTCCTL0_L	Real-Time Clock Control 0 Low	Read/write	yes	not retained	<a href="#">Section 18.3.1</a>
01h	RTCCTL0_H	Real-Time Clock Control 0 High	Read/write	n/a	not retained	<a href="#">Section 18.3.2</a>
02h	RTCCTL13	Real-Time Clock Control 1, 3	Read/write	yes	high byte retained	<a href="#">Section 18.3.3</a>
02h	RTCCTL1	Real-Time Clock Control 1	Read/write	yes	not retained	<a href="#">Section 18.3.3</a>
	or RTCCTL13_L					
03h	RTCCTL3	Real-Time Clock Control 3	Read/write	yes	retained	<a href="#">Section 18.3.4</a>
	or RTCCTL13_H					
04h	RTCOCAL	Real-Time Clock Offset Calibration	Read/write	yes	retained	<a href="#">Section 18.3.5</a>
04h	RTCOCAL_L		Read/write	yes	retained	
05h	RTCOCAL_H		Read/write	yes	retained	
06h	RTCTCMP	Real-Time Clock Temperature Compensation	Read/write	no	retained	<a href="#">Section 18.3.6</a>
06h	RTCTCMP_L		Read/write	no	retained	
07h	RTCTCMP_H		Read/write	no	retained	
08h	RTCPSC0CTL	Real-Time Prescale Timer 0 Control	Read/write	no	not retained	<a href="#">Section 18.3.27</a>
08h	RTCPSC0CTL_L		Read/write	no	not retained	
09h	RTCPSC0CTL_H		Read/write	no	not retained	
0Ah	RTCPSC1CTL	Real-Time Prescale Timer 1 Control	Read/write	no	not retained	<a href="#">Section 18.3.28</a>
0Ah	RTCPSC1CTL_L		Read/write	no	not retained	
0Bh	RTCPSC1CTL_H		Read/write	no	not retained	
0Ch	RTCPSC	Real-Time Prescale Timer 0, 1 Counter	Read/write	yes	retained	<a href="#">Section 18.3.29</a>
0Ch	RTCPSC0	Real-Time Prescale Timer 0 Counter	Read/write	yes	retained	<a href="#">Section 18.3.29</a>
	or RTCPSC_L					
0Dh	RTCPSC1	Real-Time Prescale Timer 1 Counter	Read/write	yes	retained	<a href="#">Section 18.3.30</a>
	or RTCPSC_H					
0Eh	RTCIV	Real Time Clock Interrupt Vector	Read	no	not retained	<a href="#">Section 18.3.31</a>
10h	RTCTIM0	Real-Time Clock Seconds, Minutes	Read/write	yes	retained	
10h	RTCSEC	Real-Time Clock Seconds	Read/write	yes	retained	<a href="#">Section 18.3.7</a>

**Table 18-1. RTC\_C Registers (continued)**

Offset	Acronym	Register Name	Type	Key Protected	LPM3.5 Retention	Section
or RTCTIM0_L						
11h	RTCMIN	Real-Time Clock Minutes	Read/write	yes	retained	
or RTCTIM0_H						
12h	RTCTIM1	Real-Time Clock Hour, Day of Week	Read/write	yes	retained	
12h	RTCHOUR	Real-Time Clock Hour	Read/write	yes	retained	<a href="#">Section 18.3.11</a>
or RTCTIM1_L						
13h	RTCDOW	Real-Time Clock Day of Week	Read/write	yes	retained	<a href="#">Section 18.3.13</a>
or RTCTIM1_H						
14h	RTCDATE	Real-Time Clock Date	Read/write	yes	retained	
14h	RTCDAY	Real-Time Clock Day of Month	Read/write	yes	retained	<a href="#">Section 18.3.14</a>
or RTCDATE_L						
15h	RTCMON	Real-Time Clock Month	Read/write	yes	retained	<a href="#">Section 18.3.16</a>
or RTCDATE_H						
16h	RTCYEAR	Real-Time Clock Year <sup>(1)</sup>	Read/write	yes	retained	<a href="#">Section 18.3.18</a>
18h	RTCAMINHR	Real-Time Clock Minutes, Hour Alarm	Read/write	no	retained	
18h	RTCAMIN	Real-Time Clock Minutes Alarm	Read/write	no	retained	<a href="#">Section 18.3.20</a>
or RTCAMINHR_L						
19h	RTCAHOUR	Real-Time Clock Hours Alarm	Read/write	no	retained	<a href="#">Section 18.3.22</a>
or RTCAMINHR_H						
1Ah	RTCADOWDAY	Real-Time Clock Day of Week, Day of Month Alarm	Read/write	no	retained	
1Ah	RTCADOW	Real-Time Clock Day of Week Alarm	Read/write	no	retained	<a href="#">Section 18.3.24</a>
or RTCADOWDAY_L						
1Bh	RTCADAY	Real-Time Clock Day of Month Alarm	Read/write	no	retained	<a href="#">Section 18.3.25</a>
or RTCADOWDAY_H						
1Ch	RTCBIN2BCD	Binary-to-BCD conversion register	Read/write	no	not retained	<a href="#">Section 18.3.32</a>
1Eh	RTBCD2BIN	BCD-to-binary conversion register	Read/write	no	not retained	<a href="#">Section 18.3.33</a>

<sup>(1)</sup> The year register RTCYEAR must not be accessed in byte mode.

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 18.3.1 RTCCTL0\_L Register

Real-Time Clock Control 0 Low Register

**Figure 18-3. RTCCTL0\_L Register**

7	6	5	4	3	2	1	0
RTCOFIE <sup>(1)</sup>	RTCTEVIE <sup>(1)</sup>	RTCAIE <sup>(1)</sup>	RTCRDYIE	RTCOFIG	RTCTEVIFG	RTCAIFG	RTCRDYIFG
rw-0	rw-0	rw-0	rw-0	rw-1	rw-0	rw-0	rw-0

- <sup>(1)</sup> The configuration of these bits is retained during LPM3.5 until LOCKBKUP is cleared, but not the register bits themselves; therefore, reconfiguration is required after wake-up from LPM3.5 before clearing LOCKBKUP.

**Table 18-2. RTCCTL0\_L Register Description**

Bit	Field	Type	Reset	Description
7	RTCOFIE	RW	0h	32-kHz crystal oscillator fault interrupt enable. 0b = Interrupt not enabled 1b = Interrupt enabled (LPM3/LPM3.5 wake-up enabled)
6	RTCTEVIE	RW	0h	Real-time clock time event interrupt enable. 0b = Interrupt not enabled 1b = Interrupt enabled (LPM3/LPM3.5 wake-up enabled)
5	RTCAIE	RW	0h	Real-time clock alarm interrupt enable. 0b = Interrupt not enabled 1b = Interrupt enabled (LPM3/LPM3.5 wake-up enabled)
4	RTCRDYIE	RW	0h	Real-time clock ready interrupt enable 0b = Interrupt not enabled 1b = Interrupt enabled
3	RTCOFIG	RW	1h	32-kHz crystal oscillator fault interrupt flag. This interrupt can be used as an LPM3 or LPM3.5 wake-up event. It indicates a clock failure during RTC operation. 0b = No interrupt pending 1b = Interrupt pending. A 32-kHz crystal oscillator fault occurred after the last reset.
2	RTCTEVIFG	RW	0h	Real-time clock time event interrupt flag. This interrupt can be used as an LPM3 or LPM3.5 wake-up event. 0b = No time event occurred 1b = Time event occurred
1	RTCAIFG	RW	0h	Real-time clock alarm interrupt flag. This interrupt can be used as an LPM3 or LPM3.5 wake-up event. 0b = No time event occurred 1b = Time event occurred
0	RTCRDYIFG	RW	0h	Real-time clock ready interrupt flag 0b = RTC cannot be read safely 1b = RTC can be read safely

### **18.3.2 RTCCTL0\_H Register**

Real-Time Clock Control 0 High Register

**Figure 18-4. RTCCTL0\_H Register**

7	6	5	4	3	2	1	0
RTCKEY							
rw-1	rw-0	rw-0	rw-1	rw-0	rw-1	rw-1	rw-0

**Table 18-3. RTCCTL0\_H Register Description**

Bit	Field	Type	Reset	Description
7-0	RTCKEY	RW	96h	Real-time clock key. This register should be written with A5h to unlock RTC. Any write with value other than A5h locks the module. Reads from this register always return 96h.

### 18.3.3 RTCCTL1 Register

Real-Time Clock Control Register 1

**Figure 18-5. RTCCTL1 Register**

7	6	5	4	3	2	1	0
RTCBCD	RTCHOLD <sup>(1)</sup>	RTCMODE <sup>(1)</sup>	RTCRDY	RTCSSELx <sup>(1)</sup>		RTCTEVx <sup>(1)</sup>	
rw-0	rw-1	r-1	r-1	rw-0	rw-0	rw-0	rw-0

<sup>(1)</sup> The configuration of these bits is retained during LPM3.5 until LOCKBKUP is cleared, but not the register bits themselves; therefore, reconfiguration is required after wake-up from LPM3.5 before clearing LOCKBKUP.

**Table 18-4. RTCCTL1 Register Description**

Bit	Field	Type	Reset	Description
7	RTCBCD	RW	0h	Real-time clock BCD select. Selects BCD counting for real-time clock. 0b = Binary (hexadecimal) code selected 1b = Binary coded decimal (BCD) code selected
6	RTCHOLD	RW	1h	Real-time clock hold 0b = Real-time clock is operational. 1b = When set, the calendar is stopped and the prescale counters RT0PS and RT1PS are don't care.
5	RTCMODE	R	1h	Real-time clock mode. 0b = Reserved 1b = Calendar mode. Always reads a value of 1.
4	RTCRDY	R	1h	Real-time clock ready 0b = RTC time values in transition. 1b = RTC time values safe for reading. This bit indicates when the real-time clock time values are safe for reading.
3-2	RTCSSELx	RW	0h	Real-time clock source select. 00b = BCLK 01b = Reserved. Defaults to BCLK. 10b = Reserved. Defaults to BCLK. 11b = Reserved. Defaults to BCLK
1-0	RTCTEVx	RW	0h	Real-time clock time event 00b = Minute changed 01b = Hour changed 10b = Every day at midnight (00:00) 11b = Every day at noon (12:00)

### 18.3.4 RTCCTL3 Register

Real-Time Clock Control 3 Register

**Figure 18-6. RTCCTL3 Register**

7	6	5	4	3	2	1	0
Reserved						RTCCALFx <sup>(1)</sup>	
r0	r0	r0	r0	r0	r0	rw-0	rw-0

<sup>(1)</sup> These bits are reset when the backup domain sees a hard reset and the LOCKBKUP bit is 0.

**Table 18-5. RTCCTL3 Register Description**

Bit	Field	Type	Reset	Description
7-2	Reserved	R	0h	Reserved. Always reads as 0.
1-0	RTCCALFx	RW	0h	Real-time clock calibration frequency. Selects frequency output to RTCCLK pin for calibration measurement. The corresponding port must be configured for the peripheral module function. 00b = No frequency output to RTCCLK pin 01b = 512 Hz 10b = 256 Hz 11b = 1 Hz

### 18.3.5 RTCOCAL Register

Real-Time Clock Offset Calibration Register

**Figure 18-7. RTCOCAL Register**

15	14	13	12	11	10	9	8
RTCOALS <sup>(1)</sup>	Reserved						
rw-0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
RTCOCALx <sup>(1)</sup>							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

<sup>(1)</sup> These bits are reset when the backup domain sees a hard reset and the LOCKBKUP bit is 0.

**Table 18-6. RTCOCAL Register Description**

Bit	Field	Type	Reset	Description
15	RTCOALS	RW	0h	Real-time clock offset error calibration sign. This bit decides the sign of offset error calibration. 0b = Down calibration. Frequency adjusted down. 1b = Up calibration. Frequency adjusted up.
14-8	Reserved	R	0h	Reserved. Always reads as 0.
7-0	RTCOCALx	RW	0h	Real-time clock offset error calibration. Each LSB represents approximately +1-ppm (RTCOALS = 1) or -1-ppm (RTCOALS = 0) adjustment in frequency. Maximum effective calibration value is ±240 ppm. Values written above ±240 ppm are ignored by hardware.

### 18.3.6 RTCTCMP Register

Real-Time Clock Temperature Compensation Register

**Figure 18-8. RTCTCMP Register**

15	14	13	12	11	10	9	8
RTCTCMPS <sup>(1)</sup>	RTCTCRDY <sup>(1)</sup>	RTCTCOK <sup>(1)</sup>			Reserved		
rw-0	r-1	r-0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
			RTCTCMPx <sup>(1)</sup>				
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

<sup>(1)</sup> These bits are reset when the backup domain sees a hard reset and the LOCKBKUP bit is 0.

**Table 18-7. RTCTCMP Register Description**

Bit	Field	Type	Reset	Description
15	RTCTCMPS	RW	0h	Real-time clock temperature compensation sign. This bit decides the sign of temperature compensation. <sup>(1)</sup> 0b = Down calibration. Frequency adjusted down. 1b = Up calibration. Frequency adjusted up.
14	RTCTCRDY	R	1h	Real-time clock temperature compensation ready. This is a read only bit that indicates when the RTCTCMPx can be written. Write to RTCTCMPx should be avoided when RTCTCRDY is reset.
13	RTCTCOK	R	0h	Real-time clock temperature compensation write OK. This is a read-only bit that indicates if the write to RTCTCMP is successful or not. 0b = Write to RTCTCMPx is unsuccessful 1b = Write to RTCTCMPx is successful
12-8	Reserved	R	0h	Reserved. Always reads as 0.
7-0	RTCTCMPx	RW	0h	Real-time clock temperature compensation. Value written into this register is used for temperature compensation of RTC. Each LSB represents approximately +1 ppm (RTCTCMPS = 1) or -1 ppm (RTCTCMPS = 0) adjustment in frequency. Maximum effective calibration value is ±240 ppm. Excess values written above ±240 ppm are ignored by hardware.

<sup>(1)</sup> Changing the sign-bit by writing to RTCTCMP\_H becomes effective only after also writing RTCTCMP\_L.

### 18.3.7 RTCSEC Register – Hexadecimal Format

Real-Time Clock Seconds Register – Hexadecimal Format

**Figure 18-9. RTCSEC Register**

7	6	5	4	3	2	1	0
0	Seconds						
r-0	r-0	rw	rw	rw	rw	rw	rw

**Table 18-8. RTCSEC Register Description**

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always 0
5-0	Seconds	RW	undefined	Seconds (0 to 59)

### 18.3.8 RTCSEC Register – BCD Format

Real-Time Clock Seconds Register – BCD Format

**Figure 18-10. RTCSEC Register**

7	6	5	4	3	2	1	0
0	Seconds – high digit				Seconds – low digit		
r-0	rw	rw	rw	rw	rw	rw	rw

**Table 18-9. RTCSEC Register Description**

Bit	Field	Type	Reset	Description
7	0	R	0h	Always 0
6-4	Seconds – high digit	RW	undefined	Seconds – high digit (0 to 5)
3-0	Seconds – low digit	RW	undefined	Seconds – low digit (0 to 9)

### **18.3.9 RTCMIN Register – Hexadecimal Format**

Real-Time Clock Minutes Register – Hexadecimal Format

**Figure 18-11. RTCMIN Register**

7	6	5	4	3	2	1	0
0	Minutes						
r-0	r-0	rw	rw	rw	rw	rw	rw

**Table 18-10. RTCMIN Register Description**

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always 0
5-0	Minutes	RW	undefined	Minutes (0 to 59)

### **18.3.10 RTCMIN Register – BCD Format**

Real-Time Clock Minutes Register – BCD Format

**Figure 18-12. RTCMIN Register**

7	6	5	4	3	2	1	0
0	Minutes – high digit				Minutes – low digit		
r-0	rw	rw	rw	rw	rw	rw	rw

**Table 18-11. RTCMIN Register Description**

Bit	Field	Type	Reset	Description
7	0	R	0h	Always 0
6-4	Minutes – high digit	RW	undefined	Minutes – high digit (0 to 5)
3-0	Minutes – low digit	RW	undefined	Minutes – low digit (0 to 9)

### 18.3.11 RTCHOUR Register – Hexadecimal Format

Real-Time Clock Hours Register – Hexadecimal Format

**Figure 18-13. RTCHOUR Register**

7	6	5	4	3	2	1	0
					Hours		
r-0	r-0	r-0	rw	rw	rw	rw	rw

**Table 18-12. RTCHOUR Register Description**

Bit	Field	Type	Reset	Description
7-5	0	R	0h	Always 0
4-0	Hours	RW	undefined	Hours (0 to 23)

### 18.3.12 RTCHOUR Register – BCD Format

Real-Time Clock Hours Register – BCD Format

**Figure 18-14. RTCHOUR Register**

7	6	5	4	3	2	1	0
				Hours – high digit		Hours – low digit	
r-0	r-0	rw	rw	rw	rw	rw	rw

**Table 18-13. RTCHOUR Register Description**

Bit	Field	Type	Reset	Description
7-6	0	R	0h	Always 0
5-4	Hours – high digit	RW	undefined	Hours – high digit (0 to 2)
3-0	Hours – low digit	RW	undefined	Hours – low digit (0 to 9)

### **18.3.13 RTCDOW Register**

Real-Time Clock Day of Week Register

**Figure 18-15. RTCDOW Register**

7	6	5	4	3	2	1	0		
			0	Day of week					
r-0	r-0	r-0	r-0	r-0	rw	rw	rw		

**Table 18-14. RTCDOW Register Description**

Bit	Field	Type	Reset	Description
7-3	0	R	0h	Always 0
2-0	Day of week	RW	undefined	Day of week (0 to 6)

### **18.3.14 RTCDAY Register – Hexadecimal Format**

Real-Time Clock Day of Month Register – Hexadecimal Format

**Figure 18-16. RTCDAY Register**

7	6	5	4	3	2	1	0		
			0	Day of month					
r-0	r-0	r-0	rw	rw	rw	rw	rw		

**Table 18-15. RTCDAY Register Description**

Bit	Field	Type	Reset	Description
7-5	0	R	0h	Always 0
4-0	Day of month	RW	undefined	Day of month (1 to 28, 29, 30, 31)

### **18.3.15 RTCDAY Register – BCD Format**

Real-Time Clock Day of Month Register – BCD Format

**Figure 18-17. RTCDAY Register**

7	6	5	4	3	2	1	0
			0	Day of month – high digit		Day of month – low digit	
r-0	r-0	rw	rw	rw	rw	rw	rw

**Table 18-16. RTCDAY Register Description**

Bit	Field	Type	Reset	Description
7-6	0	R	0h	
5-4	Day of month – high digit	RW	undefined	Day of month – high digit (0 to 3)
3-0	Day of month – low digit	RW	undefined	Day of month – low digit (0 to 9)

### 18.3.16 RTCMON Register – Hexadecimal Format

Real-Time Clock Month Register – Hexadecimal Format

**Figure 18-18. RTCMON Register**

7	6	5	4	3	2	1	0		
			0	Month					
r-0	r-0	r-0	r-0	rw	rw	rw	rw		

**Table 18-17. RTCMON Register Description**

Bit	Field	Type	Reset	Description
7-4	0	R	0h	Always 0
3-0	Month	RW	undefined	Month (1 to 12)

### 18.3.17 RTCMON Register – BCD Format

Real-Time Clock Month Register – BCD Format

**Figure 18-19. RTCMON Register**

7	6	5	4	3	2	1	0
			0	Month – high digit		Month – low digit	
r-0	r-0	r-0	rw	rw	rw	rw	rw

**Table 18-18. RTCMON Register Description**

Bit	Field	Type	Reset	Description
7-5	0	R	0h	Always 0
4	Month – high digit	RW	undefined	Month – high digit (0 or 1)
3-0	Month – low digit	RW	undefined	Month – low digit (0 to 9)

### 18.3.18 RTCYEAR Register – Hexadecimal Format

Real-Time Clock Year Low-Byte Register – Hexadecimal Format

**Figure 18-20. RTCYEAR Register**

15	14	13	12	11	10	9	8		
0				Year – high byte					
r-0	r-0	r-0	r-0	rw	rw	rw	rw		
7	6	5	4	3	2	1	0		
Year – low byte									
rw	rw	rw	rw	rw	rw	rw	rw		

**Table 18-19. RTCYEAR Register Description**

Bit	Field	Type	Reset	Description
15-12	0	R	0h	Always 0
11-8	Year – high byte	RW	undefined	Year – high byte. Valid values for Year are 0 to 4095.
7-0	Year – low byte	RW	undefined	Year – low byte. Valid values for Year are 0 to 4095.

### 18.3.19 RTCYEAR Register – BCD Format

Real-Time Clock Year Low-Byte Register – BCD Format

**Figure 18-21. RTCYEAR Register**

15	14	13	12	11	10	9	8
0		Century – high digit			Century – low digit		
r-0	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
Decade				Year – lowest digit			
rw	rw	rw	rw	rw	rw	rw	rw

**Table 18-20. RTCYEAR Register Description**

Bit	Field	Type	Reset	Description
15	0	R	0h	Always 0
14-10	Century – high digit	RW	undefined	Century – high digit (0 to 4)
11-8	Century – low digit	RW	undefined	Century – low digit (0 to 9)
7-4	Decade	RW	undefined	Decade (0 to 9)
3-0	Year – lowest digit	RW	undefined	Year – lowest digit (0 to 9)

### 18.3.20 RTCAMIN Register – Hexadecimal Format

Real-Time Clock Minutes Alarm Register – Hexadecimal Format

**Figure 18-22. RTCAMIN Register**

7	6	5	4	3	2	1	0
AE	0						Minutes
rw	r-0	rw	rw	rw	rw	rw	rw

**Table 18-21. RTCAMIN Register Description**

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable
6	0	R	0h	Always 0.
5-0	Minutes	RW	undefined	Minutes (0 to 59)

### 18.3.21 RTCAMIN Register – BCD Format

Real-Time Clock Minutes Alarm Register – BCD Format

**Figure 18-23. RTCAMIN Register**

7	6	5	4	3	2	1	0
AE		Minutes – high digit			Minutes – low digit		
rw	rw	rw	rw	rw	rw	rw	rw

**Table 18-22. RTCAMIN Register Description**

Bit	Field	Type	Reset	Description
7	AE	RW	0h	Alarm enable
6-4	Minutes – high digit	RW	undefined	Minutes – high digit (0 to 5)
3-0	Minutes – low digit	RW	undefined	Minutes – low digit (0 to 9)

### **18.3.22 RTCAHOUR Register – Hexadecimal Format**

Real-Time Clock Hours Alarm Register – Hexadecimal Format

**Figure 18-24. RTCAHOUR Register**

7	6	5	4	3	2	1	0
AE	0				Hours		
rw	r-0	r-0	rw	rw	rw	rw	rw

**Table 18-23. RTCAHOUR Register Description**

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable
6-5	0	R	0h	Always 0
4-0	Hours	RW	undefined	Hours (0 to 23)

### **18.3.23 RTCAHOUR Register – BCD Format**

Real-Time Clock Hours Alarm Register – BCD Format

**Figure 18-25. RTCAHOUR Register**

7	6	5	4	3	2	1	0
AE	0		Hours – high digit		Hours – low digit		
rw	r-0	rw	rw	rw	rw	rw	rw

**Table 18-24. RTCAHOUR Register Description**

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable
6	0	R	0h	Always 0
5-4	Hours – high digit	RW	undefined	Hours – high digit (0 to 2)
3-0	Hours – low digit	RW	undefined	Hours – low digit (0 to 9)

### 18.3.24 RTCADOW Register – Calendar Mode

Real-Time Clock Day of Week Alarm Register – Calendar Mode

**Figure 18-26. RTCADOW Register**

7	6	5	4	3	2	1	0
AE		0				Day of week	
rw	r-0	r-0	r-0	r-0	rw	rw	rw

**Table 18-25. RTCADOW Register Description**

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable
6-3	0	R	0h	Always 0
2-0	Day of week	RW	undefined	Day of week (0 to 6)

### 18.3.25 RTCADAY Register – Hexadecimal Format

Real-Time Clock Day of Month Alarm Register – Hexadecimal Format

**Figure 18-27. RTCADAY Register**

7	6	5	4	3	2	1	0
AE		0				Day of month	
rw	r-0	r-0	rw	rw	rw	rw	rw

**Table 18-26. RTCADAY Register Description**

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable
6-5	0	R	0h	Always 0
4-0	Day of month	RW	undefined	Day of month (1 to 28, 29, 30, 31)

### 18.3.26 RTCADAY Register – BCD Format

Real-Time Clock Day of Month Alarm Register – BCD Format

**Figure 18-28. RTCADAY Register**

7	6	5	4	3	2	1	0
AE	0	Day of month – high digit				Day of month – low digit	
rw	r-0	rw	rw	rw	rw	rw	rw

**Table 18-27. RTCADAY Register Description**

Bit	Field	Type	Reset	Description
7	AE	RW	undefined	Alarm enable
6	0	R	0h	Always 0
5-4	Day of month – high digit	RW	undefined	Day of month – high digit (0 to 3)
3-0	Day of month – low digit	RW	undefined	Day of month – low digit (0 to 9)

### **18.3.27 RTCPS0CTL Register**

Real-Time Clock Prescale Timer 0 Control Register

**Figure 18-29. RTCPS0CTL Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved				RT0IP <sup>(1)</sup>		RT0PSIE	RT0PSIFG
r0	r0	r0	rw-0	rw-0	rw-0	rw-0	rw-0

<sup>(1)</sup> The configuration of these bits is retained during LPM3.5 until LOCKBKUP is cleared, but not the register bits themselves; therefore, reconfiguration is required after wake-up from LPM3.5 before clearing LOCKBKUP.

**Table 18-28. RTCPS0CTL Register Description**

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Always reads as 0.
4-2	RT0IP	RW	0h	Prescale timer 0 interrupt interval 000b = Divide by 2 001b = Divide by 4 010b = Divide by 8 011b = Divide by 16 100b = Divide by 32 101b = Divide by 64 110b = Divide by 128 111b = Divide by 256
1	RT0PSIE	RW	0h	Prescale timer 0 interrupt enable 0b = Interrupt not enabled 1b = Interrupt enabled (LPM3/LPM3.5 wake-up enabled)
0	RT0PSIFG	RW	0h	Prescale timer 0 interrupt flag. This interrupt can be used as an LPM3 or LPM3.5 wake-up event. 0b = No time event occurred 1b = Time event occurred

### 18.3.28 RTCPS1CTL Register

Real-Time Clock Prescale Timer 1 Control Register

**Figure 18-30. RTCPS1CTL Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved				RT1IPx <sup>(1)</sup>		RT1PSIE	RT1PSIFG
r0	r0	r0	rw-0	rw-0	rw-0	rw-0	rw-0

<sup>(1)</sup> The configuration of these bits is retained during LPM3.5 until LOCKBKUP is cleared, but not the register bits themselves; therefore, reconfiguration is required after wake-up from LPM3.5 before clearing LOCKBKUP.

**Table 18-29. RTCPS1CTL Register Description**

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Always reads as 0.
4-2	RT1IPx	RW	0h	Prescale timer 1 interrupt interval 000b = Divide by 2 001b = Divide by 4 010b = Divide by 8 011b = Divide by 16 100b = Divide by 32 101b = Divide by 64 110b = Divide by 128 111b = Divide by 256
1	RT1PSIE	RW	0h	Prescale timer 1 interrupt enable 0b = Interrupt not enabled 1b = Interrupt enabled (LPM3/LPM3.5 wake-up enabled)
0	RT1PSIFG	RW	0h	Prescale timer 1 interrupt flag. This interrupt can be used as an LPM3 or LPM3.5 wake-up event. 0b = No time event occurred 1b = Time event occurred

### 18.3.29 RTCPS0 Register

Real-Time Clock Prescale Timer 0 Counter Register

**Figure 18-31. RTCPS0 Register**

7	6	5	4	3	2	1	0
RT0PS							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 18-30. RTCPS0 Register Description**

Bit	Field	Type	Reset	Description
7-0	RT0PS	RW	undefined	Prescale timer 0 counter value

### 18.3.30 RTCPS1 Register

Real-Time Clock Prescale Timer 1 Counter Register

**Figure 18-32. RTCPS1 Register**

7	6	5	4	3	2	1	0
RT1PS							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 18-31. RTCPS1 Register Description**

Bit	Field	Type	Reset	Description
7-0	RT1PS	RW	undefined	Prescale timer 1 counter value

### 18.3.31 RTCIV Register

Real-Time Clock Interrupt Vector Register

**Figure 18-33. RTCIV Register**

15	14	13	12	11	10	9	8
RTClVx							
r0	r0	r0	r0	r0	r0	r0	r0
RTClVx							
r0	r0	r0	r0	r0	r0	r0	r0

**Table 18-32. RTCIV Register Description**

Bit	Field	Type	Reset	Description
15-0	RTClVx	R	0h	Real-time clock interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: RTC oscillator failure; Interrupt Flag: RTCOIFG; Interrupt Priority: Highest 04h = Interrupt Source: RTC ready; Interrupt Flag: RTCRDYIFG 06h = Interrupt Source: RTC interval timer; Interrupt Flag: RTCTEVIFG 08h = Interrupt Source: RTC user alarm; Interrupt Flag: RTCAIFG 0Ah = Interrupt Source: RTC prescaler 0; Interrupt Flag: RT0PSIFG 0Ch = Interrupt Source: RTC prescaler 1; Interrupt Flag: RT1PSIFG 0Eh = Reserved 10h = Reserved ; Interrupt Priority: Lowest

### **18.3.32 RTCBIN2BCD Register**

Binary-to-BCD Conversion Register

**Figure 18-34. RTCBIN2BCD Register**

15	14	13	12	11	10	9	8
BIN2BCDx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
BIN2BCDx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 18-33. RTCBIN2BCD Register Description**

Bit	Field	Type	Reset	Description
15-0	BIN2BCDx	RW	0h	Read: 16-bit BCD conversion of previously written 12-bit binary number. Write: 12-bit binary number to be converted.

### **18.3.33 RTCBCD2BIN Register**

BCD-to-Binary Conversion Register

**Figure 18-35. RTCBCD2BIN Register**

15	14	13	12	11	10	9	8
BCD2BINx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
BCD2BINx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 18-34. RTCBCD2BIN Register Description**

Bit	Field	Type	Reset	Description
15-0	BCD2BINx	RW	0h	Read: 12-bit binary conversion of previously written 16-bit BCD number. Write: 16-bit BCD number to be converted.

## **Reference Module (REF\_A)**

The REF\_A module is a general purpose reference system that is used to generate voltage references required for other analog modules available on a given device such as analog-to-digital converters, digital-to-analog converters, comparators or LCD. This chapter describes the REF\_A module.

Topic	Page
<b>19.1 REF_A Introduction .....</b>	<b>657</b>
<b>19.2 Principle of Operation .....</b>	<b>658</b>
<b>19.3 REF_A Registers .....</b>	<b>660</b>

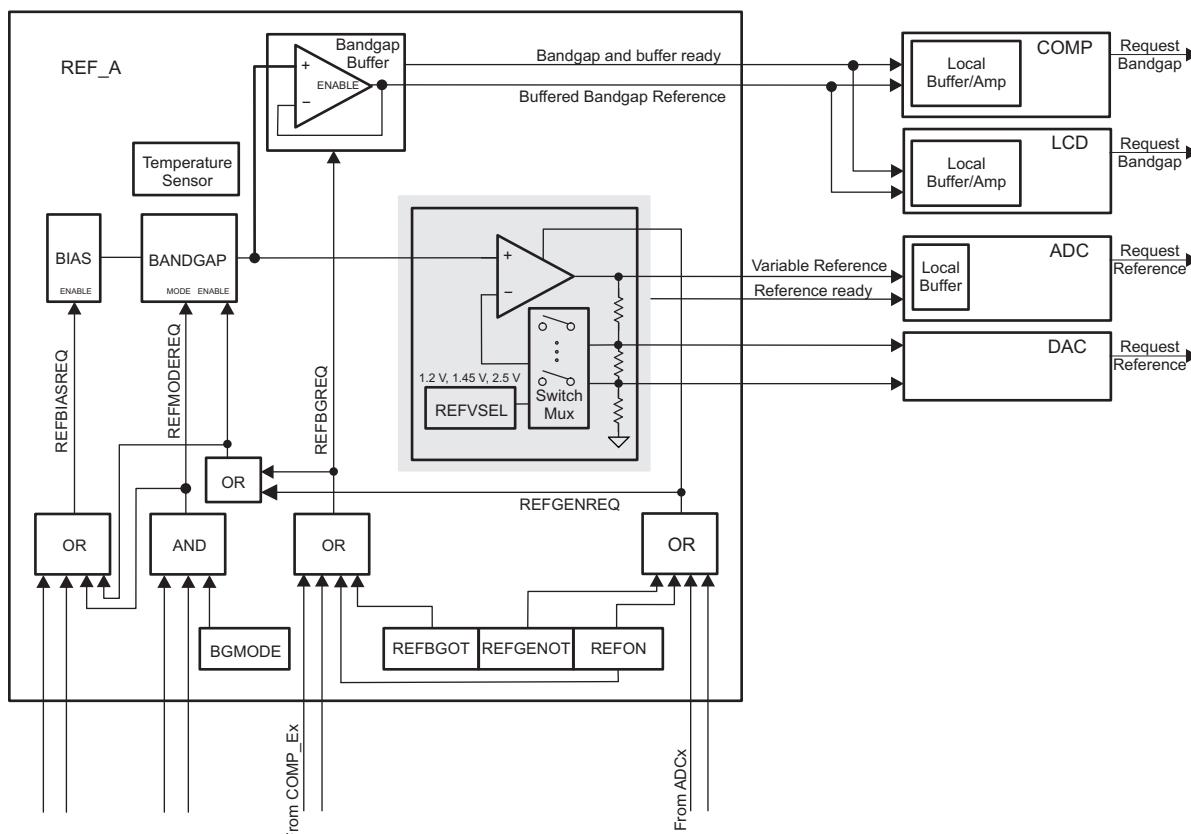
## 19.1 REF\_A Introduction

The REF\_A module is responsible for generation of all critical reference voltages for use by various analog modules in a given device. The heart of the REF\_A module is the bandgap from which all other reference voltages are derived by unity or noninverting gain stages. The REFGEN subsystem inside the REF\_A module consists of the bandgap, the bandgap bias, and the noninverting buffer stage that generates the primary voltage references available in the system: 1.2 V, 1.45 V, and 2.5 V. In addition, when enabled, a buffered bandgap voltage is available.

Features of the REF\_A include:

- Centralized factory calibrated bandgap with excellent PSRR, temperature coefficient, and accuracy
- 1.2-V, 1.45-V, or 2.5-V user-selectable internal reference voltages
- Buffered bandgap voltage available to rest of the system
- Power saving features
- Hardware reference requests and reference ready signals for bandgap and variable reference voltages for safe operation

[Figure 19-1](#) shows a block diagram of the REF\_A module in an example device with an ADC, a DAC, an LCD, and a Comparator.



**Figure 19-1. REF\_A Block Diagram**

## 19.2 Principle of Operation

The REF\_A module provides all of the necessary voltage references to be used by various analog modules on the device.

The REF\_A module contains a high-performance bandgap. This bandgap has good accuracy (factory calibrated), low temperature coefficient, and high PSRR even while operating at low power. The bandgap voltage is used to generate 1.2 V, 1.45 V, and 2.5 V reference voltages through a noninverting amplifier stage. One voltage can be selected at a time. A second output of the REF\_A module provides a buffered bandgap reference line. The REF\_A module also supports voltage references that are required for the DAC12 module, when it is available. The REF\_A module also includes the temperature sensor circuitry that operates from the bandgap. The temperature sensor is used by an ADC to measure a voltage proportional to temperature.

When the REFON bit is set to 1 in the REFCTL0 register, the bandgap, the bandgap bias, the noninverting buffer stage, and the unity gain buffer are enabled.

### 19.2.1 Low-Power Operation

The REF\_A module can support low-power applications such as LCD generation. Many of these applications do not require a very accurate reference, compared to data conversion, but low-power consumption is very important. To support these kinds of applications, the bandgap can be used in a sampled mode. In sampled mode, the bandgap circuitry is clocked by the VLO at an appropriate duty cycle. This reduces the average power of the bandgap circuitry significantly, but at the cost of accuracy. When not in sampled mode, the bandgap is in static mode. Its power is at its highest, but so is its accuracy.

Analog modules can automatically request static mode or sampled mode through their own individual request lines. In this way, each module determines which mode is appropriate for its proper operation and performance. Any one active analog module that requests static mode causes all other analog modules to use static mode, even if another analog module is requesting sampled mode. In other words, static mode request always has higher priority than sampled mode request. When the REFON bit is set, the bandgap and the bandgap bias operate in static mode.

### 19.2.2 Reference System Requests

There are three basic reference requests that are used by the REF\_A module. Each analog module can use these requests to obtain the proper response from the REF\_A module. The three basic requests are REFGENREQ, REFBGREQ, and REFMODEREQ. No interaction is required by the user code. The analog modules automatically select the proper request.

A reference request signal, REFGENREQ, is available as an input into the REFGEN subsystem. This signal represents a logical OR of individual requests coming from the various analog modules in the device that require a voltage reference to be available on the variable reference line. When a module requires a voltage reference, it asserts its corresponding REFGENREQ signal. When the REFGENREQ is asserted, the REFGEN subsystem is enabled. After the specified settling time, the variable reference line voltage is stable and ready for use. The REFVSEL settings determine which voltage is generated on the variable reference line.

After the specified settling time of the REFGEN subsystem, the REF\_A module sets the REFGENDRDY signal. This signal is used by each analog module, for example, to wait before an ADC conversion is started after a REFGENREQ was set. The generation of the reference voltage can be triggered by a timer or by software to make sure the reference voltage is ready when an analog module requires it.

In addition to the REFGENREQ, a second reference request signal, REFBGREQ is available. The REFBGREQ signal represents a logical OR of requests coming from the various analog modules that require the buffered bandgap reference line. When the REFBGREQ is asserted, the bandgap with its bias circuitry and local buffer are enabled, if it is not already enabled by a prior request.

After the specified settling time of the REFBGREQ subsystem, the REF\_A module sets the REFBGRDY signal. This signal is used by each analog module to hold operation while the buffered bandgap reference voltage is settling. The generation of the buffered bandgap voltage can be triggered by a timer or by software to make sure the reference voltage is ready when an analog module requires it.

The REFMODEREQ request signal configures the bandgap and its bias circuitry to operate in a sampled or static mode of operation. The REFMODEREQ signal represents a logical AND of individual requests coming from the various analog modules. A REFMODEREQ occurs only if at least one analog module's REFGENREQ or REFBGQ is also asserted, otherwise it is a don't care.

When REFMODEREQ = 1, the bandgap operates in sampled mode. When an analog module asserts its corresponding REFMODEREQ signal, it is requesting that the bandgap operates in sampled mode. Because REMODEREQ is a logical AND of all individual requests, any analog module requesting static mode causes the bandgap to operate in static mode. The BGMODE bit can be read as an indicator of static or sampled mode of operation.

#### 19.2.2.1 REFBGACT, REFGENACT, REFGENBUSY

Any analog module that is using the variable reference line causes REFGENACT to be set inside the REFCTL0 register. This bit is read only and indicates whether the REFGEN is active or off. Similarly, the REFBGACT is active any time one or more analog modules are actively using the buffered bandgap reference line. This bit is read only and indicates whether the REFBG is active or off.

The REFGENBUSY signal, when asserted, indicates that an analog module is using the reference and cannot have any of its settings changed. For example, during an active ADC14 conversion, the reference voltage level should not be changed. REFGENBUSY is asserted when there is an active ADC14 conversion. When it is asserted, REFGENBUSY write protects the REFCTL0 register. This prevents the reference from being disabled or its level changed during any active conversion.

#### 19.2.2.2 ADC14

For devices that contain an ADC14 module, there are two buffers. Refer to ADC14 block diagram in the ADC14 chapter. The large buffer (BUF\_EXT) can be used to drive the reference voltage, present on the variable reference line, external to the device. This buffer has larger power consumption due to a selectable burst mode as well as its need to drive larger DC loads that may be present outside the device. The large buffer is enabled continuously when REFON = 1, REFOUT = 1, and ADC14REFBURST = 0. When ADC14REFBURST = 1, the large buffer is enabled only during an ADC conversion, shutting down automatically upon completion of a conversion to save power. In addition, when REFON = 1 and REFOUT = 1, the second small buffer (BUF\_INT) is automatically disabled. In this case, the output of the large buffer is connected to the capacitor array through an internal analog switch. This ensures the same reference is used throughout the system. If REFON = 1 and REFOUT = 0, the small buffer is used for ADC conversion and the large buffer remains disabled. The small buffer can operate in burst mode as well by setting ADC14REFBURST = 1.

## 19.3 REF\_A Registers

The REF\_A registers are listed in [Table 19-1](#). The base address can be found in the device-specific data sheet. The address offset is listed in [Table 19-1](#).

---

**NOTE:** All registers have half-word or byte access. For a generic register ANYREG, the suffix "\_L" (ANYREG\_L) refers to the lower byte of the register (bits 0 through 7). The suffix "\_H" (ANYREG\_H) refers to the upper byte of the register (bits 8 through 15).

---

**Table 19-1. REF\_A Registers**

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	REFCTL0	REFCTL0	Read/write	Word	0008h	<a href="#">Section 19.3.1</a>
00h		REFCTL0_L	Read/write	Byte	08h	
01h		REFCTL0_H	Read/write	Byte	00h	

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 19.3.1 REFCTL0 Register (offset = 00h) [reset = 0008h]

REF Control Register 0

**Figure 19-2. REFCTL0 Register**

15	14	13	12	11	10	9	8
Reserved		REFBGRDY	REFGENRDY	BGMODE	REFGENBUSY	REFBGACT	REFGENACT
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
REFBGOT	REFGENOT	REFVSEL		REFTCOFF	Reserved	REFOUT	REFON
rw-0	rw-0	rw-0	rw-0	rw-1	rw-0	rw-0	rw-0
Can be modified only when REFGENBUSY = 0							

**Table 19-2. REFCTL0 Register Description**

Bit	Field	Type	Reset	Description
15-14	Reserved	R	0h	Reserved. Always reads as 0.
13	REFBGRDY	R	0h	Buffered bandgap voltage ready status. 0b = Buffered bandgap voltage is not ready to be used. 1b = Buffered bandgap voltage is ready to be used.
12	REFGENRDY	R	0h	Variable reference voltage ready status. 0b = Reference voltage output is not ready to be used. 1b = Reference voltage output is ready to be used.
11	BGMODE	R	0h	Bandgap mode. Read only. 0b = Static mode 1b = Sampled mode
10	REFGENBUSY	R	0h	Reference generator busy. Read only. 0b = Reference generator not busy 1b = Reference generator busy
9	REFBGACT	R	0h	Reference bandgap active. Read only. 0b = Reference bandgap buffer not active 1b = Reference bandgap buffer active
8	REFGENACT	R	0h	Reference generator active. Read only. 0b = Reference generator not active 1b = Reference generator active
7	REFBGOT	RW	0h	Bandgap and bandgap buffer one-time trigger. If written with a "1" the generation of the buffered bandgap voltage is started. Once the bandgap buffer voltage request is set this bit is cleared by hardware. 0b = No trigger 1b = Generation of the bandgap voltage is started by writing 1 or by a hardware trigger.
6	REFGENOT	RW	0h	Reference generator one-time trigger. If written with a "1" the generation of the variable reference voltage is started. Once the reference voltage request is set this bit is cleared by hardware. 0b = No trigger 1b = Generation of the reference voltage is started by writing 1 or by a hardware trigger.
5-4	REFVSEL	RW	0h	Reference voltage level select. Can be modified only when REFGENBUSY = 0. 00b = 1.2 V available when reference requested or REFON = 1 01b = 1.45 V available when reference requested or REFON = 1 10b = Reserved 11b = 2.5 V available when reference requested or REFON = 1
3	REFTCOFF	RW	1h	Temperature sensor disabled. Can be modified only when REFGENBUSY = 0. 0b = Temperature sensor enabled 1b = Temperature sensor disabled to save power

**Table 19-2. REFCTL0 Register Description (continued)**

Bit	Field	Type	Reset	Description
2	Reserved	RW	0h	Reserved.
1	REFOUT	RW	0h	Reference output buffer. Can be modified only when REFGENBUSY = 0. 0b = Reference output not available externally. 1b = Reference output available externally. If ADC14REFBURST = 0, output is available continuously. If ADC14REFBURST = 1, output is available only during an ADC14 conversion.
0	REFON	RW	0h	Reference enable. Can be modified only when REFGENBUSY = 0. 0b = Disables reference if no other reference requests are pending. 1b = Enables reference in static mode.

**ADC14**

The ADC14 module is a high-performance 14-bit analog-to-digital converter (ADC). This chapter describes the operation of the ADC14 module.

Topic	Page
<b>20.1 ADC14 Introduction .....</b>	<b>664</b>
<b>20.2 ADC14 Operation.....</b>	<b>665</b>
<b>20.3 ADC14 Registers .....</b>	<b>680</b>

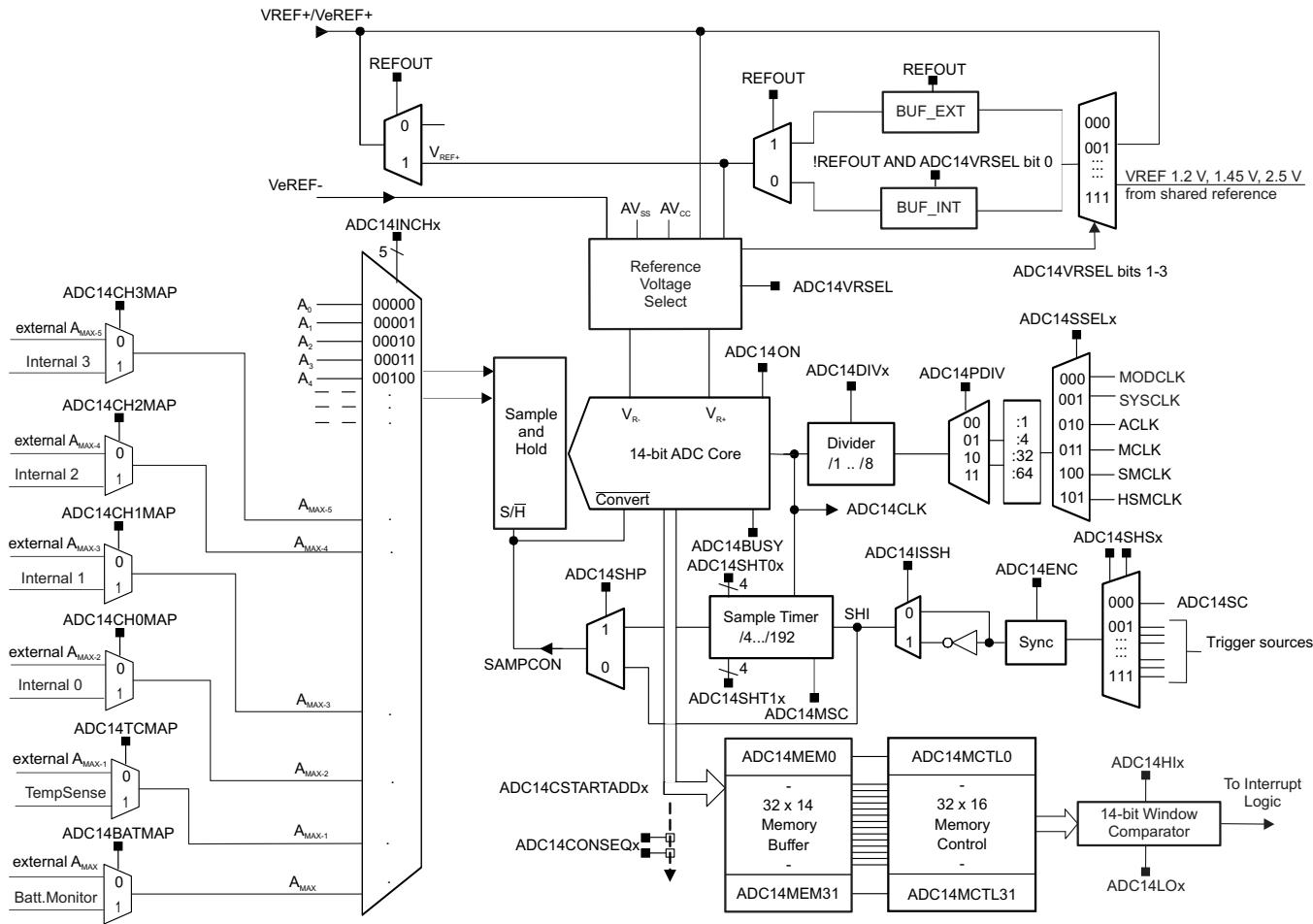
## 20.1 ADC14 Introduction

The ADC14 module supports fast 14-bit analog-to-digital conversions. The module implements a 14-bit SAR core, sample select control, and up to 32 independent conversion-and-control buffers. The conversion-and-control buffer allows up to 32 independent analog-to-digital converter (ADC) samples to be converted and stored without any CPU intervention.

ADC14 features include:

- 1-Msps maximum conversion rate at maximum resolution of 14 bits
- Monotonic 14-bit converter with no missing codes
- Sample-and-hold with programmable sampling periods controlled by software or timers
- Conversion initiation by software or timers
- Software-selectable on-chip reference voltage generation (1.2 V, 1.45 V, or 2.5 V) with option to make available externally
- Software-selectable internal or external reference
- Up to 32 individually configurable external input channels, single-ended or differential input selection available
- Internal conversion channels for internal temperature sensor and  $\frac{1}{2} \times AV_{CC}$  and four more internal channels available on select devices (see device data sheet for availability and function)
- Independent channel-selectable reference sources for positive reference
- Selectable conversion clock source
- Single-channel, repeat-single-channel, sequence (autoscan), and repeat-sequence (repeated autoscan) conversion modes
- Interrupt vector register for fast decoding of 38 ADC interrupts
- 32 conversion-result storage registers
- Window comparator for low power monitoring of input signals of conversion-result registers

[Figure 20-1](#) shows the block diagram of ADC14. The reference generation is located in the reference module (REF).



Copyright © 2016, Texas Instruments Incorporated

**Figure 20-1. ADC14 Block Diagram**

**NOTE:** MODCLK and SYSCLK are from MODOSC and SYSOSC respectively. See the *Clock System (CS)* chapter for more information.

See the device-specific data sheet for trigger sources available.

See the device-specific data sheet for availability and functionality of internal channels 0–3.

See the device-specific data sheet for the total number of ADC14 external input channels.

REFOUT bit is present in the REF module register.

When ADC14VRSEL(0) = 1 and REFOUT = 0, the internal reference buffer BUF\_INT is enabled and used.

When ADC14VRSEL(0) = 1 and REFOUT = 1, the external reference buffer BUF\_EXT is enabled and used.

## 20.2 ADC14 Operation

The ADC14 module is configured with user software. The setup and operation of the ADC14 is discussed in the following sections.

## 20.2.1 14-Bit ADC Core

The ADC core converts an analog input to its 14-bit digital representation. The core uses two programmable/selectable voltage levels ( $V_{R+}$  and  $V_{R-}$ ) to define the upper and lower limits of the conversion. The digital output ( $N_{ADC}$ ) is full scale (3FFFh) when the input signal is equal to or higher than  $V_{R+}$ , and zero when the input signal is equal to or lower than  $V_{R-}$ . The input channel and the reference voltage levels ( $V_{R+}$  and  $V_{R-}$ ) are defined in the conversion-control memory.

[Equation 5](#) shows the conversion formula for the ADC result  $N_{ADC}$  for single-ended mode.

$$N_{ADC} = 16384 \times \frac{V_{in+} - V_{R-}}{V_{R+} - V_{R-}}, 1 \text{ LSB} = \frac{V_{R+} - V_{R-}}{16384} \quad (5)$$

[Equation 6](#) shows the conversion formula for the ADC result  $N_{ADC}$  for differential mode.

$$N_{ADC} = \left( 8192 \times \frac{V_{in+} - V_{in-}}{V_{R+} - V_{R-}} \right) + 8192, 1 \text{ LSB} = \frac{V_{R+} - V_{R-}}{8192} \quad (6)$$

[Equation 7](#) describes the input voltage at which the ADC output saturates for singled-ended mode.

$$V_{in+} = V_{R+} - V_{R-} - 1 \text{ LSB} \quad (7)$$

[Equation 8](#) describes the input voltage at which the ADC output saturates for differential mode.

$$V_{in+} - V_{in-} = V_{R+} - V_{R-} - 1 \text{ LSB} \quad (8)$$

The ADC14 core is configured by two control registers, ADC14CTL0 and ADC14CTL1. The core is reset when ADC14ON = 0. When ADC14ON = 1, reset is removed and the core is ready to power up when a valid conversion is triggered. The ADC14 can be turned off when not in use to save power. If during a conversion the ADC14ON bit is set to 0, the conversion is abruptly exited and everything is powered down. With few exceptions, the ADC14 control bits can be modified only when ADC14ENC = 0. ADC14ENC must be set to 1 before any conversion can take place.

The conversion results are always stored in binary unsigned format. For differential inputs, this means that the result has an offset of 8192 added to it to make the number positive. The data format bit ADC14DF in ADC14CTL1 allows the user to read the conversion results as binary unsigned or signed binary (2s complement).

### 20.2.1.1 Conversion Clock Selection

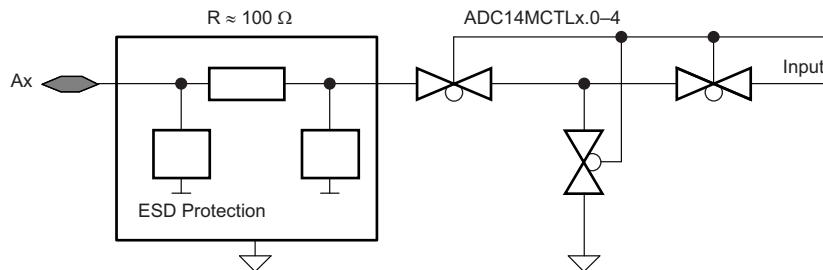
The ADC14CLK is used both as the conversion clock and to generate the sampling period when the pulse sampling mode is selected. The ADC14 source clock is selected using the ADC14SSELx bit. The input clock can be dived by 1, 4, 32, or 64 using the ADC14PDIV bits and then subsequently divided by 1–8 using the ADC14DIV bits. Possible ADC14CLK sources are MODCLK, SYSCLK, ACLK, MCLK, SMCLK, and HSMCLK.

The user must ensure that the clock chosen for ADC14CLK remains active until the end of a conversion. If the clock is removed during a conversion, the operation does not complete and any result is invalid.

### 20.2.2 ADC14 Inputs and Multiplexer

Up to 32 external and up to 6 internal analog signals are selected as the channel for conversion by the analog input multiplexer. The number of channels available is device specific and given in the device-specific data sheet. The input multiplexer is a break-before-make type to reduce input-to-input noise injection resulting from channel switching (see [Figure 20-2](#)). The input multiplexer is also a T-switch to minimize the coupling between channels. Channels that are not selected are isolated from the A/D and the intermediate node is connected to analog ground ( $AV_{ss}$ ), so that the stray capacitance is grounded to eliminate crosstalk.

The ADC14 uses the charge redistribution method. When the inputs are internally switched, the switching action may cause transients on the input signal. These transients decay and settle before causing errant conversion.



**Figure 20-2. Analog Multiplexer**

#### 20.2.2.1 Analog Port Selection

The ADC14 inputs are multiplexed with digital port pins. When analog signals are applied to digital gates, parasitic current can flow from  $V_{CC}$  to GND. This parasitic current occurs if the input voltage is near the transition level of the gate. Disabling the digital part of the port pin eliminates the parasitic current flow and reduces overall current consumption. The PySELx bits provide the ability to disable the port pin input and output buffers.

#### 20.2.3 Voltage References

The ADC14 module may use an on-chip shared reference module that supplies three selectable voltage levels of 1.2 V, 1.45 V, and 2.5 V (see the REF module chapter for configuration details) to supply  $V_{R+}$  and  $V_{R-}$ . These reference voltages may be used internally and externally on pin VREF+. Alternatively, external references may be supplied for  $V_{R+}$  and  $V_{R-}$  through pins VREF+/VeREF+ and VeREF-, respectively. TI recommends connecting VeREF- to onboard ground when using ADC14VRSEL settings 1110b or 1111b.

---

**NOTE:** The maximum sampling rate of ADC14 is limited to 200 ksp/s when the internal reference is used together with BUF\_EXT (ADC14VRSEL = 0001b and REFOUT = 1). In all other reference settings the ADC14 sampling rate can be up to 1 Msps.

---

#### 20.2.4 Auto Power Down

The ADC14 is designed for low-power applications. When the ADC14 is not actively converting, the core is automatically disabled and automatically reenabled when needed. The MODOSC or SYSOSC are also automatically enabled to provide MODCLK or SYSCLK to ADC14 when needed and disabled when not needed for ADC14 or for rest of the device.

#### 20.2.5 Power Modes

The ADC14 supports two power modes selected through ADC14PWRMD bits in ADC14CTL1 register. ADC14PWRMD = 00b selects regular-power mode, and ADC14PWRMD = 10b selects low-power mode. ADC14 supports 8-, 10-, 12-, and 14-bit resolution settings selected through the ADC14RES bits in the ADC14CTL1 register.

The regular-power mode (ADC14PWRMD = 00b) supports sampling rates up to 1 Msps and can be used with any of the resolutions settings. The low-power mode (ADC14PWRMD = 10b) is a power saving mode recommended for 12-, 10-, or 8-bit resolution settings with sampling rates not exceeding 200 ksp/s.

#### 20.2.6 Sample and Conversion Timing

An analog-to-digital conversion is initiated with a rising edge of the sample input signal SHI. The source for SHI is selected with the SHSx bits and includes the following:

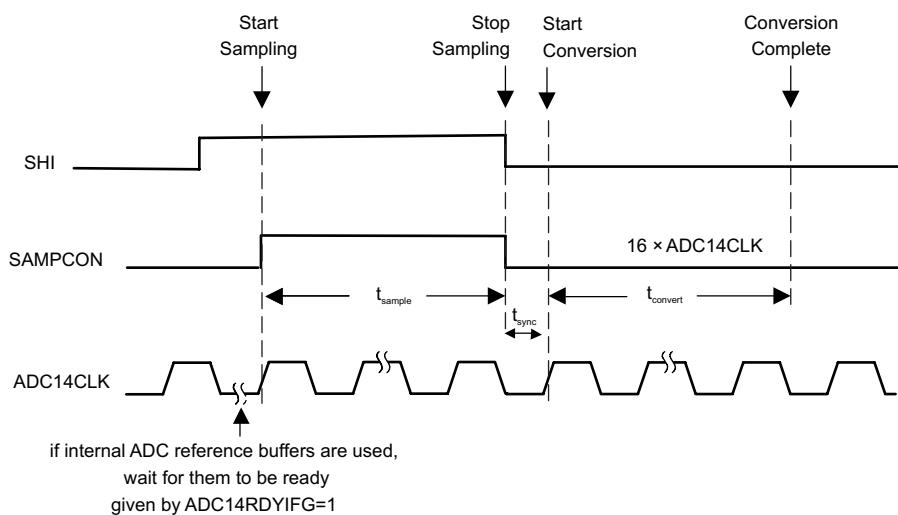
- ADC14SC bit

- Up to seven other sources which may include timer output (see to the device-specific data sheet for available sources).

The analog-to-digital conversion requires 9, 11, 14 and 16 ADC14CLK cycles for 8-bit, 10-bit, 12-bit, and 14-bit resolution modes respectively. The polarity of the SHI signal source can be inverted with the ADC14ISSH bit. The SAMPCON signal controls the sample period and start of conversion. When SAMPCON is high, sampling is active. The high-to-low SAMPON transition starts the analog-to-digital conversion. Two different sample-timing methods are defined by control bit ADC14SHP, extended sample mode, and pulse mode. See the device-specific data sheet for available timers for SHI sources.

#### 20.2.6.1 Extended Sample Mode

The extended sample mode is selected when ADC14SHP = 0. The SHI signal directly controls SAMPON and defines the length of the sample period  $t_{sample}$ . If an ADC internal buffer is used, the user should assert the sample trigger, wait for the ADC14RDYIFG flag to be set (indicating the ADC14 local buffered reference is settled) and then keep the sample trigger asserted for the desired sample period before de-asserting. Alternately, if an internal ADC buffer is used the user may assert the sample trigger for the desired sample time plus the max time for the reference and buffers to settle (reference and buffer settling times are provided in the device-specific data sheet). The maximum sampling time must not exceed 420  $\mu$ s. An ADC internal buffer is used when ADC14VRSEL= 0001 or 1111. When SAMPON is high, sampling is active. The high-to-low SAMPON transition starts the conversion after synchronization with ADC14CLK (see [Figure 20-3](#)).

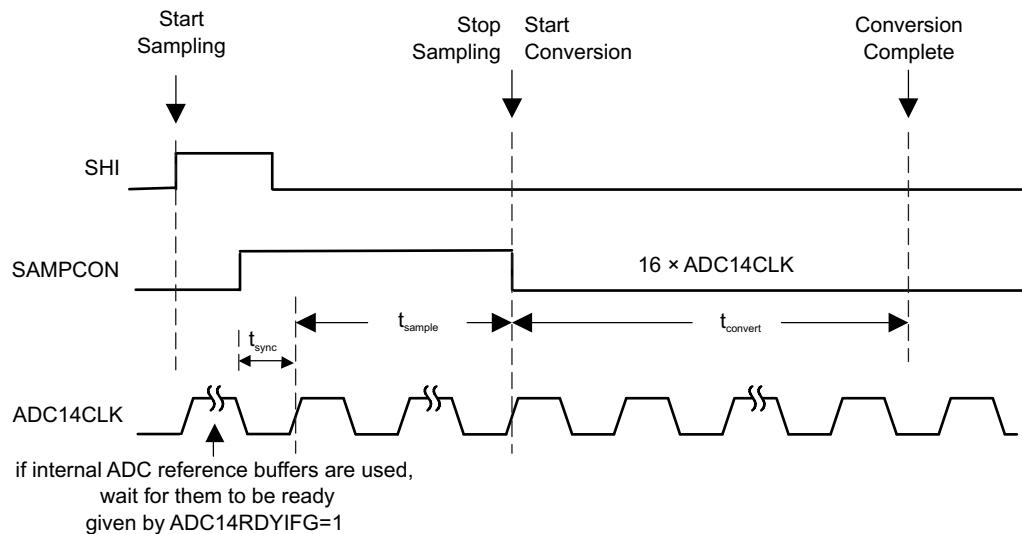


**Figure 20-3. Extended Sample Mode in 14-Bit Mode**

#### 20.2.6.2 Pulse Sample Mode

The pulse sample mode is selected when ADC14SHP = 1. The SHI signal is used to trigger the sampling timer. The ADC14SHT0x and ADC14SHT1x bits in ADC14CTL0 control the interval of the sampling timer that defines the SAMPON sample period  $t_{sample}$ . The sampling timer keeps SAMPON high while waiting for reference and internal buffer to settle (if the internal reference is used), synchronization with ADC14CLK and for the programmed interval  $t_{sample}$ . The total sampling time is  $t_{sample}$  plus the time for ADC14RDYIFG to go high (if the ADC internal reference buffers are used) plus  $t_{sync}$ , where  $t_{sync}$  is the time to sync to the ADC14CLK (see [Figure 20-4](#)).

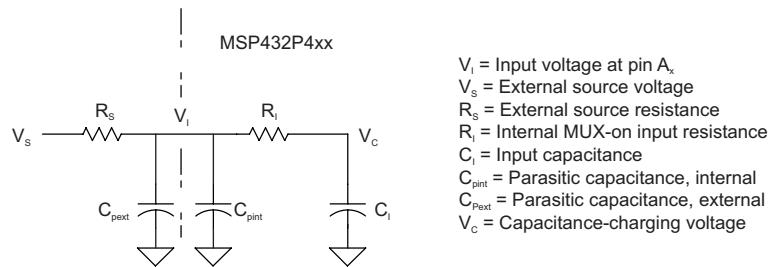
The ADC14SHTx bits select the sampling time in 4x multiples of ADC14CLK. The programmable range of sampling timer is 4 to 192 ADC14CLK cycles. ADC14SHT0x selects the sampling time for ADC14MCTL8 to ADC14MCTL23, and ADC14SHT1x selects the sampling time for ADC14MCTL0 to ADC14MCTL7 and ADC14MCTL24 to ADC14MCTL31.



**Figure 20-4. Pulse Sample Mode in 14-Bit Mode**

#### 20.2.6.3 Sample Timing Considerations

When SAMPCON = 0, all  $A_x$  inputs are high impedance. When SAMPCON = 1, the selected  $A_x$  input can be modeled as an RC low-pass filter during the sampling time  $t_{sample}$  (see Figure 20-5). An internal MUX-on input resistance  $R_i$  (see device-specific data sheet) in series with capacitor  $C_i$  (see device-specific data sheet) is seen by the source. The capacitor  $C_i$  voltage  $V_c$  must be charged to within one-half LSB of the source voltage  $V_s$  for an accurate n-bit conversion, where n is the bits of resolution required.



**Figure 20-5. Analog Input Equivalent Circuit**

The resistance of the source  $R_s$  and  $R_i$  affect  $t_{sample}$ . Use Equation 5 to calculate the minimum sampling time  $t_{sample}$  for a n-bit conversion, where n equals the bits of resolution.

$$t_{sample} \geq (n+1) \times \ln(2) \times [(R_s + R_i) \times C_i + R_s \times (C_{pext} + C_{pint})], R_s < 100k\Omega \quad (9)$$

See the device-specific data sheet for  $R_i$ ,  $C_i$ , and  $C_{pint}$  values.  $C_{pint}$  value is specified in the data sheet as part of Digital Inputs electrical specification.

Consider the example below for minimum sample time calculation for a 14-bit Analog-to-Digital conversion.

$$R_i = 1k\Omega, C_i = 15pF, C_{pint} = 5pF, n = 14$$

$$R_s = 10k\Omega, C_{pext} = 10pF$$

Substituting these values into Equation 5, the minimum sample time required would be 3.28  $\mu$ s.

### 20.2.7 Conversion Memory

There are 32 ADC14MEMx conversion memory registers to store conversion results. Each ADC14MEMx is configured with an associated ADC14MCTLx control register. The ADC14VRSEL bits define the voltage reference and the ADC14INCHx and ADC14DIF bits select the input channels. The ADC14EOS bit defines the end of sequence when a sequential conversion mode is used. A sequence rolls over from ADC14MEM31 to ADC14MEM0 when the ADC14EOS bit in ADC14MCTL31 is not set.

The CSTARTADDx bits define the first ADC14MCTLx used for any conversion. If the conversion mode is single-channel or repeat-single-channel, the CSTARTADDx points to the single ADC14MCTLx to be used.

If the conversion mode selected is either sequence-of-channels or repeat-sequence-of-channels, CSTARTADDx points to the first ADC14MCTLx location to be used in a sequence. A pointer, not visible to software, is incremented automatically to the next ADC14MCTLx in a sequence when each conversion completes. The sequence continues until an ADC14EOS bit in ADC14MCTLx is processed; this is the last control byte processed.

When conversion results are written to a selected ADC14MEMx, the corresponding flag in the ADC14IFGRx register is set.

[Table 20-1](#) summarizes the possible conversion results.

**Table 20-1. ADC14 Conversion Result Formats**

Analog Input Voltage Range	ADC14DIF	ADC14DF	ADC14RES	Ideal Conversion Results (With Offset Added When ADC14DIF = 1)	ADC14MEMx Read Value
$V_{in} - V_R:$ $-V_{REF}$ to $+V_{REF}$	0	0	00	0 to 255	0000h to 00FFh
	0	0	01	0 to 1023	0000h to 03FFh
	0	0	10	0 to 4095	0000h to 0FFFh
	0	0	11	0 to 16383	0000h to 3FFFh
	0	1	00	-128 to 127	8000h to 7F00h
	0	1	01	-512 to 511	8000h to 7FC0h
	0	1	10	-2048 to 2047	8000h to 7FF0h
	0	1	11	-8192 to 8191	8000h to 7FFCh
	1	0	00	-128 to 127 (0 to 255)	0000h to 00FFh
$V_{in+} - V_{in-}:$ $-V_{REF}$ to $+V_{REF}$	1	0	01	-512 to 511 (0 to 1023)	0000h to 03FFh
	1	0	10	-2048 to 2047 (0 to 4095)	0000h to 0FFFh
	1	0	11	-8192 to 8192 (0 to 16383)	0000h to 3FFFh
	1	1	00	-128 to 127	8000h to 7F00h
	1	1	01	-512 to 511	8000h to 7FC0h
	1	1	10	-2048 to 2047	8000h to 7FF0h
	1	1	11	-8192 to 8191	8000h to 7FFCh

### 20.2.8 ADC14 Conversion Modes

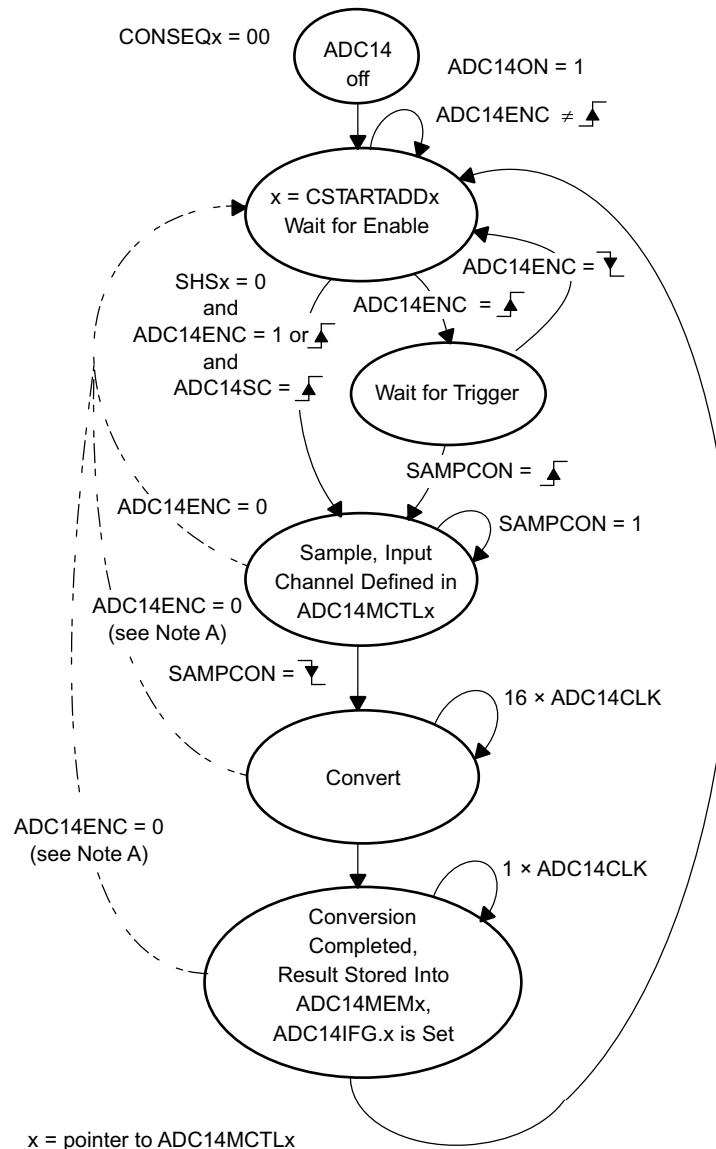
Table 20-2 lists the four ADC14 operating modes that are selected by the CONSEQx bits. All state diagrams assume a 14-bit resolution setting.

**Table 20-2. Conversion Mode Summary**

ADC14CONSEQx	Mode	Operation	Section
00	Single-channel single-conversion	A single channel is converted once.	<a href="#">Section 20.2.8.1</a>
01	Sequence-of-channels (autoscan)	A sequence of channels is converted once.	<a href="#">Section 20.2.8.2</a>
10	Repeat-single-channel	A single channel is converted repeatedly.	<a href="#">Section 20.2.8.3</a>
11	Repeat-sequence-of-channels (repeated autoscan)	A sequence of channels is converted repeatedly.	<a href="#">Section 20.2.8.4</a>

### 20.2.8.1 Single-Channel Single-Conversion Mode

A single channel is sampled and converted once. The ADC result is written to the ADC14MEMx defined by the CSTARTADDx bits. [Figure 20-6](#) shows the flow of the single-channel single-conversion mode when ADC14RES = 03h for 14-bit mode. When ADC14SC triggers a conversion, successive conversions can be triggered by the ADC14SC bit. When any other trigger source is used, ADC14ENC must be toggled between each conversion. ADC14ENC low pulse duration must be at least three ADC14CLK cycles.

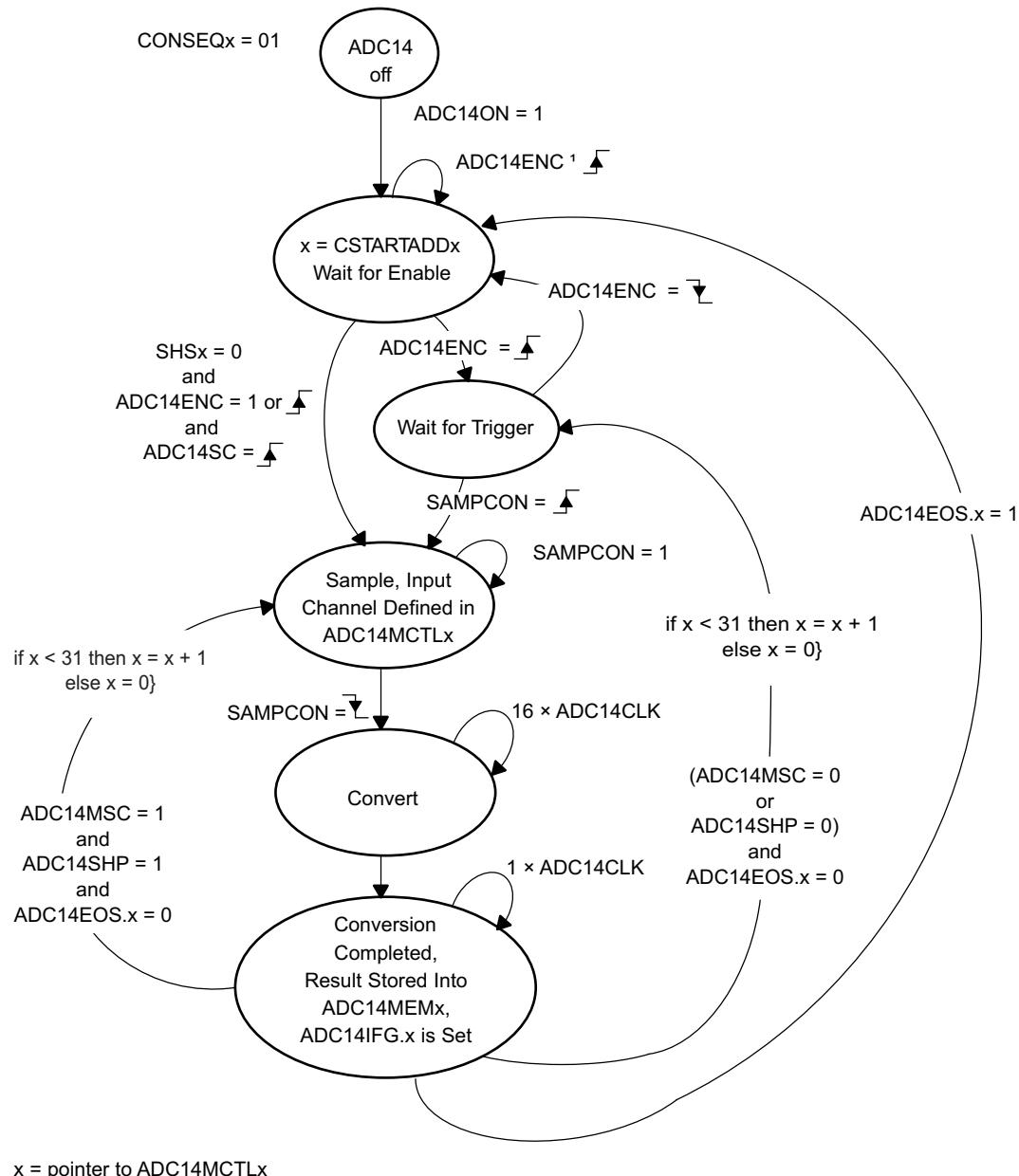


A Conversion result is unpredictable.

**Figure 20-6. Single-Channel Single-Conversion Mode**

### 20.2.8.2 Sequence-of-Channels Mode (Autoscan Mode)

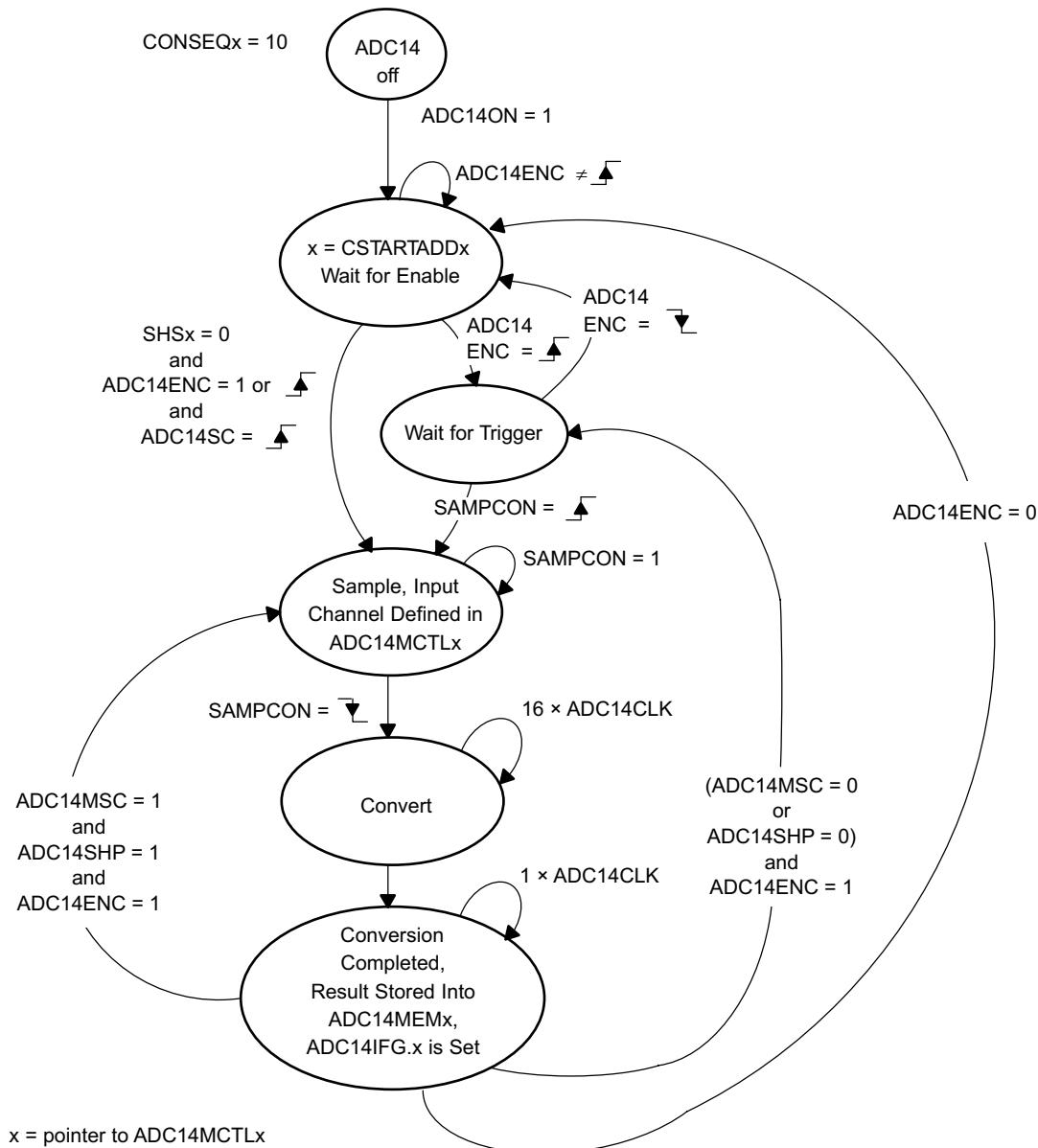
In sequence-of-channels mode, also referred to as autoscan mode, a sequence of channels is sampled and converted once. The ADC results are written to the conversion memories starting with the ADCMEMX defined by the CSTARTADDx bits. The sequence stops after the measurement of the channel with a set ADC14EOS bit. Figure 20-7 shows the sequence-of-channels mode when ADC14RES = 03h for 14-bit mode. When ADC14SC starts a sequence, additional sequences can also be started by the ADC14SC bit. When any other trigger source is used to start a sequence, ADC14ENC must be toggled between each sequence. ADC14ENC low pulse duration must be at least three ADC14CLK cycles.



**Figure 20-7. Sequence-of-Channels Mode**

### 20.2.8.3 Repeat-Single-Channel Mode

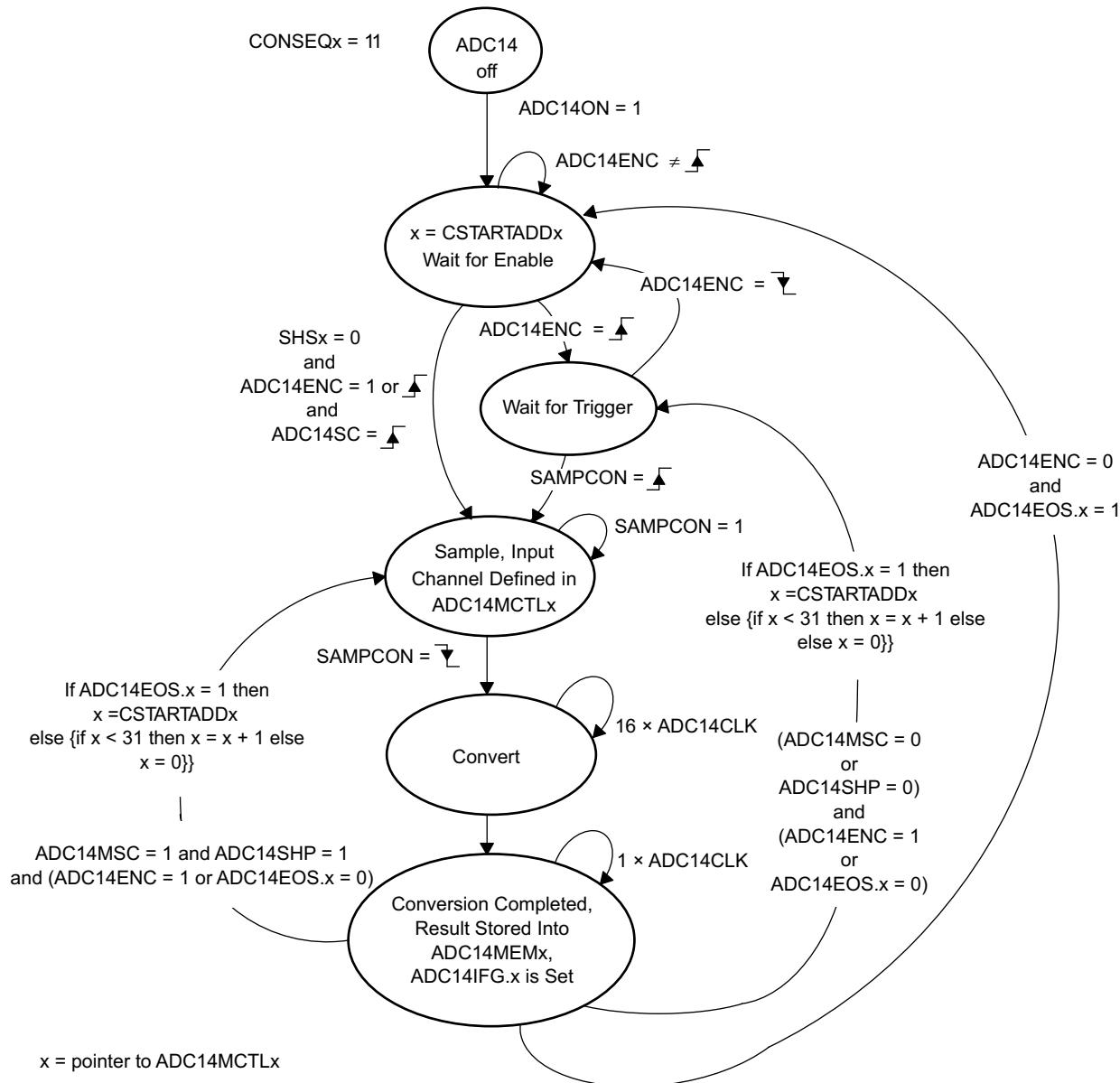
A single channel is sampled and converted continuously. The ADC results are written to the ADC14MEMx defined by the CSTARTADDx bits. It is necessary to read the result after the completed conversion because only one ADC14MEMx memory is used and is overwritten by the next conversion. Figure 20-8 shows the repeat-single-channel mode when ADC14RES = 03h for 14-bit mode. ADC14ENC low pulse duration must be at least three ADC14CLK cycles.



**Figure 20-8. Repeat-Single-Channel Mode**

#### 20.2.8.4 Repeat-Sequence-of-Channels Mode (Repeated Autoscan Mode)

In this mode, a sequence of channels is sampled and converted repeatedly. This mode is also referred to as repeated autoscan mode. The ADC results are written to the conversion memories starting with the ADC14MEMx defined by the CSTARTADDx bits. The sequence ends after the measurement of the channel with a set ADC14EOS bit and the next trigger signal restarts the sequence. Figure 20-9 shows the repeat-sequence-of-channels mode when ADC14RES = 03h for 14-bit mode. ADC14ENC low pulse duration must be at least three ADC14CLK cycles.



**Figure 20-9. Repeat-Sequence-of-Channels Mode**

### 20.2.8.5 Using the Multiple Sample and Convert (ADC14MSC) Bit

To configure the converter to perform successive conversions automatically and as quickly as possible, a multiple sample and convert function is available. When ADC14MSC = 1, CONSEQx > 0, and the sample timer is used, the first rising edge of the SHI signal triggers the first conversion. Successive conversions are triggered automatically as soon as the prior conversion is completed. Additional rising edges on SHI are ignored until the sequence is completed in the single-sequence mode, or until the ADC14ENC bit is toggled in repeat-single-channel or repeated-sequence modes. The function of the ADC14ENC bit is unchanged when using the ADC14MSC bit.

### 20.2.8.6 Stopping Conversions

Stopping ADC14 activity depends on the mode of operation. The recommended ways to stop an active conversion or conversion sequence are:

- Reset ADC14ENC in single-channel single-conversion mode to stop a conversion immediately. The results are unreliable. For reliable results, poll the busy bit until it is reset before clearing ADC14ENC.
- Reset ADC14ENC during repeat-single-channel operation to stop the converter at the end of the current conversion.
- Reset ADC14ENC during a sequence or repeat-sequence mode to stop the converter at the end of the current conversion.
- To stop conversion immediately in any mode, set CONSEQx = 0 and reset the ADC14ENC bit. In this case, conversion data are unreliable.

---

**NOTE: No ADC14EOS bit set for sequence**

If no ADC14EOS bit is set and a sequence mode is selected, resetting the ADC14ENC bit does not stop the sequence. To stop the sequence, first select a single-channel mode and then reset ADC14ENC.

---

### 20.2.9 Window Comparator

The window comparator allows to monitor analog signals without any CPU interaction. It is enabled for the desired ADC14MEMx conversion with the ADC14WINC bit in the ADC14MCTLx register. The window comparator interrupts are:

- The ADC14LO interrupt flag (ADC14LOIFG) is set if the current result of the ADC14 conversion is less than the low threshold defined in register ADC14LO.
- The ADC14HI interrupt flag (ADC14HIIIGH) is set if the current result of the ADC14 conversion is greater than the high threshold defined in the register ADC14HI.
- The ADC14IN interrupt flag (ADC14INIFG) is set if the current result of the ADC14 conversion is greater than or equal to the low threshold defined in register ADC14LO and less than or equal to the high threshold defined in the register ADC14HI.

These interrupts are generated independent of the conversion mode. The update of the window comparator interrupt flags happens after the ADC14IFGx.

There are two sets of window comparator threshold registers ADC14LO0, ADC14HI0 and ADC14LO1, ADC14HI1. The ADC14WINCTH bit in the Conversion Memory Control Register (ADC14MCTLx) selects between the two sets of window comparator threshold registers. When ADC14WINCTH is set to 0, ADC14LO0 and ADC14HI0 threshold registers are selected and when ADC14WINCTH is set to 1, ADC14LO1 and ADC14HI1 threshold registers are selected for memory conversion x.

The lower and higher threshold in the ADC14LOx and ADC14HIx registers must be set in the correct data format. If the binary unsigned data format is selected by ADC14DF = 0, then the thresholds in the registers ADC14LOx and ADC14HIx must be written as binary unsigned values. If the signed binary (2s complement) data format is selected by ADC14DF = 1, then the thresholds in the registers ADC14LOx and ADC14HIx must be written as signed binary (2s complement). Changing the ADC14DF bit or the ADC14RES bits reset the threshold registers.

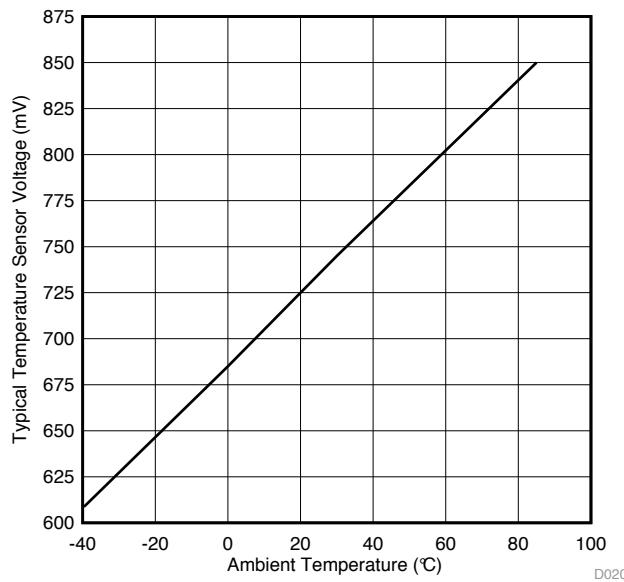
The interrupt flags are reset by the user software. The ADC14 sets the interrupt flags each time a new conversion result is available in the ADC14MEMx register if applicable. Interrupt flags are not cleared by hardware. The user software resets the window comparator interrupt flags according to the application requirements.

### **20.2.10 Using the Integrated Temperature Sensor**

To use the on-chip temperature sensor, enable the temperature sensor input channel by setting the ADC14TCMAP bit to 1 in the ADC14CTL1 register. Select the analog input channel ADC14INCHx = MAX–1, where MAX is the maximum number of external ADC input channels for the device starting count at zero, for the temperature sensor. Any other configuration is done as if that external channel was selected, including reference selection and conversion memory selection. The temperature sensor is in the REF module.

Figure 20-10 shows a typical temperature sensor transfer function. The transfer function shown here is only an example. Calibration is required to determine the corresponding voltages for the specific device. When using the temperature sensor, the sample period must be greater than 5  $\mu$ s. The temperature sensor offset error can be large and must be calibrated for most applications. Temperature calibration values are available for use in the TLV descriptors (see the device-specific data sheet for locations).

REFON bit must be set to 1 in REF module for using the temperature sensor. The reference choices for converting the temperature sensor are the same as with any other ADC channel.



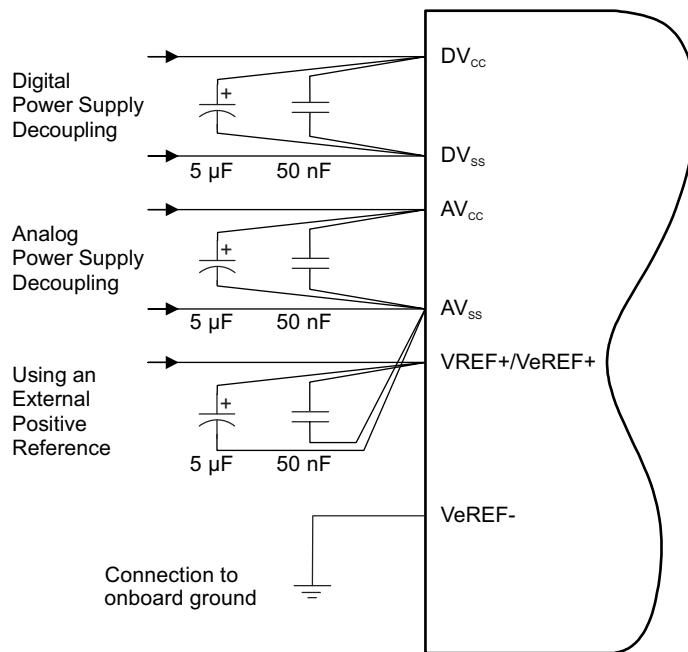
**Figure 20-10. Typical Temperature Sensor Transfer Function**

### 20.2.11 ADC14 Grounding and Noise Considerations

As with any high-resolution ADC, appropriate printed circuit board layout and grounding techniques should be followed to eliminate ground loops, unwanted parasitic effects, and noise.

Ground loops are formed when return current from the ADC flows through paths that are common with other analog or digital circuitry. If care is not taken, this current can generate small unwanted offset voltages that can add to or subtract from the reference or input voltages of the ADC. Figure 20-11 shows connections that prevent ground loops.

In addition to grounding, ripple and noise spikes on the power-supply lines that are caused by digital switching or switching power supplies can corrupt the conversion result. A noise-free design using separate analog and digital ground planes with a single-point connection is recommended to achieve high accuracy.



**Figure 20-11. ADC14 Grounding and Noise Considerations**

### 20.2.12 ADC14 Calibration

The device TLV structure contains calibration values that can be used to improve the measurement capability of the ADC14. Refer to [Section 4.9](#) of the System Controller (SYSCTL) chapter for more details.

### 20.2.13 ADC14 Interrupts

The ADC14 has the following interrupt sources:

- ADC14IFG0 to ADC14IFG31

The ADC14IFGx bits are set when their corresponding ADC14MEMx memory register is loaded with a conversion result. An interrupt request is generated if the corresponding ADC14IE bit is set and the interrupt registers in ARM® Cortex™-M4 and NVIC are configured properly. If an interrupt flag is already set when the corresponding interrupt is enabled, then an interrupt request is generated. The conversion result written into ADC14MEMx result register also sets the ADC14LOIFG, ADC14INIFG, or ADC14HIIFG if applicable.

- ADC14OV: ADC14MEMx overflow

The ADC14OV condition occurs when a conversion result is written to any ADC14MEMx before its previous conversion result was read.

- ADC14TOV: ADC14 conversion time overflow

The ADC14TOV condition is generated when another sample-and-conversion is requested before the current conversion is completed. The DMA is triggered after the conversion in single-channel conversion mode or after the completion of a sequence of channel conversions in sequence-of-channels conversion mode.

- ADC14LOIFG, ADC14INIFG, and ADC14HIIFG for ADC14MEMx
- ADC14RDYIFG: ADC14 local buffered reference ready

The ADC14RDYIFG is set when the ADC14 local buffered reference is ready. It can be used during extended sample mode instead of adding the maximum ADC14 local buffered reference settle time to the sample signal time.

#### 20.2.13.1 ADC14IV, Interrupt Vector Generator

All ADC14 interrupt sources are prioritized and combined to source a single interrupt vector. The interrupt vector register ADC14IV is used to determine which enabled ADC14 interrupt source requested an interrupt.

The highest-priority enabled ADC14 interrupt generates a number in the ADC14IV register (see register description). This number can be evaluated or added to the program counter (PC) to automatically enter the appropriate software routine. Disabled ADC14 interrupts do not affect the ADC14IV value.

Read access of the ADC14IV register automatically resets the highest-pending Interrupt condition and flag except the ADC14IFGx flags. ADC14IFGx bits are reset automatically by accessing their associated ADC14MEMx register or may be reset with software.

Write access to the ADC14IV register clears all pending Interrupt conditions and flags.

If another interrupt is pending after servicing of an interrupt, another interrupt is generated. For example, if the ADC14OV and ADC14IFG3 interrupts are pending when the interrupt service routine accesses the ADC14IV register, the ADC14OV interrupt condition is reset automatically. After the ADC14OV interrupt service is completed, the ADC14IFG3 generates another interrupt.

## 20.3 ADC14 Registers

[Table 20-3](#) shows the ADC14 registers and the address offset of each register. Refer to the device-specific data sheet for the base address of the module.

**Table 20-3. ADC14 Registers**

Offset	Acronym	Register Name	Type	Reset	Section
000h	ADC14CTL0	Control 0 Register	Read/write	00000000h	<a href="#">Section 20.3.1</a>
004h	ADC14CTL1	Control 1 Register	Read/write	00000030h	<a href="#">Section 20.3.2</a>
008h	ADC14LO0	Window Comparator Low Threshold 0 Register	Read/write	00000000h	<a href="#">Section 20.3.3</a>
00Ch	ADC14HI0	Window Comparator High Threshold 0 Register	Read/write	00003FFFh	<a href="#">Section 20.3.4</a>
010h	ADC14LO1	Window Comparator Low Threshold 1 Register	Read/write	00000000h	<a href="#">Section 20.3.5</a>
014h	ADC14HI1	Window Comparator High Threshold 1 Register	Read/write	00003FFFh	<a href="#">Section 20.3.6</a>
018h to 094h	ADC14MCTL0 to ADC14MCTL31	Memory Control 0 to Memory Control 31 Register	Read/write	00000000h	<a href="#">Section 20.3.7</a>
098h to 114h	ADC14MEM0 to ADC14MEM31	Memory 0 to Memory 31 Register	Read/write	undefined	<a href="#">Section 20.3.8</a>
13Ch	ADC14IER0	Interrupt Enable 0 Register	Read/write	00000000h	<a href="#">Section 20.3.9</a>
140h	ADC14IER1	Interrupt Enable 1 Register	Read/write	00000000h	<a href="#">Section 20.3.10</a>
144h	ADC14IFGR0	Interrupt Flag 0 Register	Read	00000000h	<a href="#">Section 20.3.11</a>
148h	ADC14IFGR1	Interrupt Flag 1 Register	Read	00000000h	<a href="#">Section 20.3.12</a>
14Ch	ADC14CLRIFGR0	Clear Interrupt Flag 0 Register	Write	00000000h	<a href="#">Section 20.3.13</a>
150h	ADC14CLRIFGR1	Clear Interrupt Flag 1 Register	Write	00000000h	<a href="#">Section 20.3.14</a>
154h	ADC14IV	Interrupt Vector Register	Read	00000000h	<a href="#">Section 20.3.15</a>

---

**NOTE:** This is a 32-bit module and can be accessed through word (32-bit) or half-word (16-bit) or byte (8-bit) accesses.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### **20.3.1 ADC14CTL0 Register (offset = 00h) [reset = 00000000h]**

## ADC14 Control 0 Register

**Figure 20-12. ADC14CTL0 Register**

**Table 20-4. ADC14CTL0 Register Description**

Bit	Field	Type	Reset	Description
31-30	ADC14PDIV	RW	0h	ADC14 predivider. This bit predivides the selected ADC14 clock source. Can be modified only when ADC14ENC = 0. 00b = Predivide by 1 01b = Predivide by 4 10b = Predivide by 32 11b = Predivide by 64
29-27	ADC14SHSx	RW	0h	ADC14 sample-and-hold source select. Can be modified only when ADC14ENC = 0. 000b = ADC14SC bit 001b = See device-specific data sheet for source 010b = See device-specific data sheet for source 011b = See device-specific data sheet for source 100b = See device-specific data sheet for source 101b = See device-specific data sheet for source 110b = See device-specific data sheet for source 111b = See device-specific data sheet for source
26	ADC14SHP	RW	0h	ADC14 sample-and-hold pulse-mode select. This bit selects the source of the sampling signal (SAMPCON) to be either the output of the sampling timer or the sample-input signal directly. Can be modified only when ADC14ENC = 0. 0b = SAMPCON signal is sourced from the sample-input signal. 1b = SAMPCON signal is sourced from the sampling timer.
25	ADC14ISSH	RW	0h	ADC14 invert signal sample-and-hold. Can be modified only when ADC14ENC = 0. 0b = The sample-input signal is not inverted. 1b = The sample-input signal is inverted. Setting ADC14ISSH = 1 and triggering a conversion using ADC14SC is not recommended. ADC14SC bit gets reset to 0 automatically at the end of conversion and if ADC14ISSH = 1, the 1->0 transition on ADC14SC triggers another conversion.

**Table 20-4. ADC14CTL0 Register Description (continued)**

Bit	Field	Type	Reset	Description
24-22	ADC14DIVx	RW	0h	ADC14 clock divider. Can be modified only when ADC14ENC = 0. 000b = /1 001b = /2 010b = /3 011b = /4 100b = /5 101b = /6 110b = /7 111b = /8
21-19	ADC14SSELx	RW	0h	ADC14 clock source select. Can be modified only when ADC14ENC = 0. 000b = MODCLK 001b = SYSCLK 010b = ACLK 011b = MCLK 100b = SMCLK 101b = HSMCLK 110b = Reserved 111b = Reserved
18-17	ADC14CONSEQx	RW	0h	ADC14 conversion sequence mode select Can be modified only when ADC14ENC = 0. 00b = Single-channel, single-conversion 01b = Sequence-of-channels 10b = Repeat-single-channel 11b = Repeat-sequence-of-channels
16	ADC14BUSY	R	0h	ADC14 busy. This bit indicates an active sample or conversion operation. 0b = No operation is active. 1b = A sequence, sample, or conversion is active.
15-12	ADC14SHT1x	RW	0h	ADC14 sample-and-hold time. These bits define the number of ADC14CLK cycles in the sampling period for registers ADC14MEM8 to ADC14MEM23. Can be modified only when ADC14ENC = 0. 0000b = 4 0001b = 8 0010b = 16 0011b = 32 0100b = 64 0101b = 96 0110b = 128 0111b = 192 1000b to 1111b = Reserved
11-8	ADC14SHT0x	RW	0h	ADC14 sample-and-hold time. These bits define the number of ADC14CLK cycles in the sampling period for registers ADC14MEM0 to ADC14MEM7 and ADC14MEM24 to ADC14MEM31. Can be modified only when ADC14ENC = 0. 0000b = 4 0001b = 8 0010b = 16 0011b = 32 0100b = 64 0101b = 96 0110b = 128 0111b = 192 1000b to 1111b = Reserved

**Table 20-4. ADC14CTL0 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
7	ADC14MSC	RW	0h	ADC14 multiple sample and conversion. Valid only for sequence or repeated modes. 0b = The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-convert. 1b = The first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed.
6-5	Reserved	R	0h	Reserved. Always reads as 0.
4	ADC14ON	RW	0h	ADC14 on 0b = ADC14 off 1b = ADC14 on. ADC core is ready to power up when a valid conversion is triggered.
3-2	Reserved	R	0h	Reserved. Always reads as 0.
1	ADC14ENC	RW	0h	ADC14 enable conversion 0b = ADC14 disabled 1b = ADC14 enabled ADC14ENC low pulse width must be at-least 3 ADC14CLK cycles.
0	ADC14SC	RW	0h	ADC14 start conversion. Software-controlled sample-and-conversion start. ADC14SC and ADC14ENC may be set together with one instruction. ADC14SC is reset automatically. 0b = No sample-and-conversion-start 1b = Start sample-and-conversion

**20.3.2 ADC14CTL1 Register (offset = 04h) [reset = 00000030h]**

ADC14 Control 1 Register

**Figure 20-13. ADC14CTL1 Register**

31	30	29	28	27	26	25	24
		Reserved				ADC14CH3MAP	ADC14CH2MAP
r-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0
23	22	21	20	19	18	17	16
ADC14TCMAP	ADC14BATMAP	Reserved	ADC14CSTARTADDx				
rw-0	rw-0	r-0	rw-0	rw-0	rw-0	rw-0	rw-0
15	14	13	12	11	10	9	8
		Reserved					
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved	ADC14RES		ADC14DF	ADC14REFBURST	ADC14PWRMD		
r-0	r-0	rw-1	rw-1	rw-0	rw-0	rw-0	rw-0
	Can be modified only when ADC14ENC = 0						

**Table 20-5. ADC14CTL1 Register Description**

Bit	Field	Type	Reset	Description
31-28	Reserved	R	0h	Reserved. Always reads as 0.
27	ADC14CH3MAP	RW	0h	Controls internal channel 3 selection to ADC input channel MAX – 5 0b = ADC input channel internal 3 is not selected 1b = ADC input channel internal 3 is selected for ADC input channel MAX – 5
26	ADC14CH2MAP	RW	0h	Controls internal channel 2 selection to ADC input channel MAX – 4 0b = ADC input channel internal 2 is not selected 1b = ADC input channel internal 2 is selected for ADC input channel MAX – 4
25	ADC14CH1MAP	RW	0h	Controls internal channel 1 selection to ADC input channel MAX – 3 0b = ADC input channel internal 1 is not selected 1b = ADC input channel internal 1 is selected for ADC input channel MAX – 3
24	ADC14CH0MAP	RW	0h	Controls internal channel 0 selection to ADC input channel MAX – 2 0b = ADC input channel internal 0 is not selected 1b = ADC input channel internal 0 is selected for ADC input channel MAX – 2
23	ADC14TCMAP	RW	0h	Controls temperature sensor ADC input channel selection 0b = ADC internal temperature sensor channel is not selected for ADC 1b = ADC internal temperature sensor channel is selected for ADC input channel MAX – 1
22	ADC14BATMAP	RW	0h	Controls 1/2 AVCC ADC input channel selection 0b = ADC internal 1/2 x AVCC channel is not selected for ADC 1b = ADC internal 1/2 x AVCC channel is selected for ADC input channel MAX
21	Reserved	R	0h	Reserved. Always reads as 0.
20-16	ADC14CSTARTADDx	RW	0h	ADC14 conversion start address. These bits select which ADC14 conversion memory register is used for a single conversion or for the first conversion in a sequence. The value of CSTARTADDx is 0h to 1Fh, corresponding to ADC14MEM0 to ADC14MEM31
15-6	Reserved	R	0h	Reserved. Always reads as 0.

**Table 20-5. ADC14CTL1 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
5-4	ADC14RES	RW	3h	ADC14 resolution. This bit defines the conversion result resolution. Can be modified only when ADC14ENC = 0. 00b = 8 bit (9 clock cycle conversion time) 01b = 10 bit (11 clock cycle conversion time) 10b = 12 bit (14 clock cycle conversion time) 11b = 14 bit (16 clock cycle conversion time)
3	ADC14DF	RW	0h	ADC14 data read-back format. Data is always stored in the binary unsigned format. 0b = Binary unsigned. Theoretically, for ADC14DIF = 0 and 14-bit mode, the analog input voltage - V(REF) results in 0000h, and the analog input voltage + V(REF) results in 3FFFh. 1b = Signed binary (2s complement), left aligned. Theoretically, for ADC14DIF = 0 and 14-bit mode, the analog input voltage - V(REF) results in 8000h, and the analog input voltage + V(REF) results in 7FFCh.
2	ADC14REFBURST	RW	0h	ADC reference buffer burst. Can be modified only when ADC14ENC = 0. 0b = ADC reference buffer on continuously 1b = ADC reference buffer on only during sample-and-conversion
1-0	ADC14PWRMD	RW	0h	ADC power modes. Can be modified only when ADC14ENC = 0. 00b = Regular-power mode for use with any resolution setting. Sample rate can be up to 1 Msps. 01b = Reserved 10b = Low-power mode for 12-bit, 10-bit, and 8-bit resolution settings. Sample rate must not exceed 200 ksp.s. 11b = Reserved

### 20.3.3 ADC14LO0 Register (offset = 08h) [reset = 00000000h]

ADC14 Window Comparator Low Threshold 0 Register

The data format that is used to write and read ADC14LO0 depends on the value of the ADC14DF bit in the ADC14CTL1 register. If ADC14DF = 0, the data is binary unsigned and right aligned. If ADC14DF = 1, the data is 2s complement and left aligned. Refer to the ADC14LO0 bit field description for details.

**Figure 20-14. ADC14LO0 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
ADC14LO0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
ADC14LO0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 20-6. ADC14LO0 Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads as 0.
15-0	ADC14LO0	RW	0h	<p>Low threshold 0. Can be modified only when ADC14ENC = 0.</p> <p>If ADC14DF = 0, unsigned binary format:</p> <p>The 14-bit threshold value must be right aligned. Bit 13 is the MSB. Bits 15-14 are 0 in 14-bit mode, bits 15-12 are 0 in 12-bit mode, bits 15-10 are 0 in 10-bit mode, and bits 15-8 are 0 in 8-bit mode.</p> <p>The reset value is: 0h</p> <p>If ADC14DF = 1, 2s-complement format:</p> <p>The 14-bit threshold value must be left aligned. Bit 15 is the MSB. Bits 1-0 are 0 in 14-bit mode, bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode.</p> <p>The reset value is: 8000h</p>

### 20.3.4 ADC14HI0 Register (offset = 0Ch) [reset = 00003FFFh]

ADC14 Window Comparator High Threshold 0 Register

The data format that is used to write and read ADC14HI0 depends on the value of the ADC14DF bit in the ADC14CTL1 register. If ADC14DF = 0, the data is binary unsigned and right aligned. If ADC14DF = 1, the data is 2s complement and left aligned. Refer to the ADC14HI0 bit field description for details.

**Figure 20-15. ADC14HI0 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
ADC14HI0							
rw-0	rw-0	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
ADC14HI0							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 20-7. ADC14HI0 Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads as 0.
15-0	ADC14HI0	RW	3FFFh	<p>High threshold 0. Can be modified only when ADC14ENC = 0.</p> <p>If ADC14DF = 0, unsigned binary format:</p> <p>The 14-bit threshold value must be right aligned. Bit 13 is the MSB. Bits 15-14 are 0 in 14-bit mode, bits 15-12 are 0 in 12-bit mode, bits 15-10 are 0 in 10-bit mode, and bits 15-8 are 0 in 8-bit mode.</p> <p>The reset value is: 3FFFh (14 bit), 0FFFh (12 bit), 03FFh (10 bit), or 00FFh (8 bit)</p> <p>If ADC14DF = 1, 2s-complement format:</p> <p>The 14-bit threshold value must be left aligned. Bit 15 is the MSB. Bits 1-0 are 0 in 14-bit mode, bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode.</p> <p>The reset value is: 7FFCh (14 bit), 7FF0h (12 bit), 7FC0h (10 bit), or 7F00h (8 bit)</p>

### 20.3.5 ADC14LO1 Register (offset = 10h) [reset = 00000000h]

ADC14 Window Comparator Low Threshold 1 Register

The data format that is used to write and read ADC14LO1 depends on the value of the ADC14DF bit in the ADC14CTL1 register. If ADC14DF = 0, the data is binary unsigned and right aligned. If ADC14DF = 1, the data is 2s complement and left aligned. Refer to the ADC14LO1 bit field description for details.

**Figure 20-16. ADC14LO1 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
ADC14LO1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
ADC14LO1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 20-8. ADC14LO1 Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads as 0.
15-0	ADC14LO1	RW	0h	<p>Low threshold 1. Can be modified only when ADC14ENC = 0.</p> <p>If ADC14DF = 0, unsigned binary format:</p> <p>The 14-bit threshold value must be right aligned. Bit 13 is the MSB. Bits 15-14 are 0 in 14-bit mode, bits 15-12 are 0 in 12-bit mode, bits 15-10 are 0 in 10-bit mode, and bits 15-8 are 0 in 8-bit mode.</p> <p>The reset value is: 0h</p> <p>If ADC14DF = 1, 2s-complement format:</p> <p>The 14-bit threshold value must be left aligned. Bit 15 is the MSB. Bits 1-0 are 0 in 14-bit mode, bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode.</p> <p>The reset value is: 8000h</p>

### 20.3.6 ADC14HI1 Register (offset = 14h) [reset = 00003FFFh]

ADC14 Window Comparator High Threshold 1 Register

The data format that is used to write and read ADC14HI1 depends on the value of the ADC14DF bit in the ADC14CTL1 register. If ADC14DF = 0, the data is binary unsigned and right aligned. If ADC14DF = 1, the data is 2s complement and left aligned. Refer to the ADC14HI1 bit field description for details.

**Figure 20-17. ADC14HI1 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
ADC14HI1							
rw-0	rw-0	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
7	6	5	4	3	2	1	0
ADC14HI1							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1

**Table 20-9. ADC14HI1 Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads as 0.
15-0	ADC14HI1	RW	3FFFh	High threshold 1. Can be modified only when ADC14ENC = 0. If ADC14DF = 0, unsigned binary format: The 14-bit threshold value must be right aligned. Bit 13 is the MSB. Bits 15-14 are 0 in 14-bit mode, bits 15-12 are 0 in 12-bit mode, bits 15-10 are 0 in 10-bit mode, and bits 15-8 are 0 in 8-bit mode. The reset value is: 3FFFh (14 bit), 0FFFh (12 bit), 03FFh (10 bit), or 00FFh (8 bit) If ADC14DF = 1, 2s-complement format: The 14-bit threshold value must be left aligned. Bit 15 is the MSB. Bits 1-0 are 0 in 14-bit mode, bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode. The reset value is: 7FFCh (14 bit), 7FF0h (12 bit), 7FC0h (10 bit), or 7F00h (8 bit)

**20.3.7 ADC14MCTL0 to ADC14MCTL31 Register (offset = 018h to 094h) [reset = 00000000h]**

ADC14 Conversion Memory Control x Register (x = 0 to 31)

**Figure 20-18. ADC14MCTL0 to ADC14MCTL31 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
23	22	21	20	19	18	17	16
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
ADC14WINCTH	ADC14WINC	ADC14DIF	Reserved	ADC14VRSEL			
rw-0	rw-0	rw-0	r-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
ADC14EOS	Reserved		ADC14INCHx				
rw-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0	rw-0
	Can be modified only when ADC14ENC = 0						

**Table 20-10. ADC14MCTL0 to ADC14MCTL31 Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	Reserved. Always reads as 0.
15	ADC14WINCTH	RW	0h	Window comparator threshold register selection Can be modified only when ADC14ENC = 0. 0b = Use window comparator thresholds 0, ADC14LO0 and ADC14HI0 1b = Use window comparator thresholds 1, ADC14LO1 and ADC14HI1
14	ADC14WINC	RW	0h	Comparator window enable Can be modified only when ADC14ENC = 0. 0b = Comparator window disabled 1b = Comparator window enabled
13	ADC14DIF	RW	0h	Differential mode. Can be modified only when ADC14ENC = 0. 0b = Single-ended mode enabled 1b = Differential mode enabled
12	Reserved	R	0h	Reserved. Always reads as 0.
11-8	ADC14VRSEL	RW	0h	Selects combinations of V(R+) and V(R-) sources as well as the buffer selection and buffer on or off. When REFOUT = 1, VeREF+ buffered configuration is not available. Can be modified only when ADC14ENC = 0. 0000b = V(R+) = AVCC, V(R-) = AVSS 0001b = V(R+) = VREF buffered, V(R-) = AVSS 0010b to 1101b = Reserved 1110b = V(R+) = VeREF+, V(R-) = VeREF- 1111b = V(R+) = VeREF+ buffered, V(R-) = VeREF- It is recommended to connect VeREF- to on-board ground when VeREF- is selected for V(R-).
7	ADC14EOS	RW	0h	End of sequence. Indicates the last conversion in a sequence. Can be modified only when ADC14ENC = 0. 0b = Not end of sequence 1b = End of sequence
6-5	Reserved	R	0h	Reserved. Always reads as 0.

**Table 20-10. ADC14MCTL0 to ADC14MCTL31 Register Description (continued)**

Bit	Field	Type	Reset	Description
4-0	ADC14INCHx	RW	0h	<p>Input channel select. If even channels are set as differential then odd channel configuration is ignored.</p> <p>Can be modified only when ADC14ENC = 0.</p> <p>00000b = If ADC14DIF = 0: A0; If ADC14DIF = 1: Ain+ = A0, Ain- = A1</p> <p>00001b = If ADC14DIF = 0: A1; If ADC14DIF = 1: Ain+ = A0, Ain- = A1</p> <p>00010b = If ADC14DIF = 0: A2; If ADC14DIF = 1: Ain+ = A2, Ain- = A3</p> <p>00011b = If ADC14DIF = 0: A3; If ADC14DIF = 1: Ain+ = A2, Ain- = A3</p> <p>00100b = If ADC14DIF = 0: A4; If ADC14DIF = 1: Ain+ = A4, Ain- = A5</p> <p>00101b = If ADC14DIF = 0: A5; If ADC14DIF = 1: Ain+ = A4, Ain- = A5</p> <p>00110b = If ADC14DIF = 0: A6; If ADC14DIF = 1: Ain+ = A6, Ain- = A7</p> <p>00111b = If ADC14DIF = 0: A7; If ADC14DIF = 1: Ain+ = A6, Ain- = A7</p> <p>01000b = If ADC14DIF = 0: A8; If ADC14DIF = 1: Ain+ = A8, Ain- = A9</p> <p>01001b = If ADC14DIF = 0: A9; If ADC14DIF = 1: Ain+ = A8, Ain- = A9</p> <p>01010b = If ADC14DIF = 0: A10; If ADC14DIF = 1: Ain+ = A10, Ain- = A11</p> <p>01011b = If ADC14DIF = 0: A11; If ADC14DIF = 1: Ain+ = A10, Ain- = A11</p> <p>01100b = If ADC14DIF = 0: A12; If ADC14DIF = 1: Ain+ = A12, Ain- = A13</p> <p>01101b = If ADC14DIF = 0: A13; If ADC14DIF = 1: Ain+ = A12, Ain- = A13</p> <p>01110b = If ADC14DIF = 0: A14; If ADC14DIF = 1: Ain+ = A14, Ain- = A15</p> <p>01111b = If ADC14DIF = 0: A15; If ADC14DIF = 1: Ain+ = A14, Ain- = A15</p> <p>10000b = If ADC14DIF = 0: A16; If ADC14DIF = 1: Ain+ = A16, Ain- = A17</p> <p>10001b = If ADC14DIF = 0: A17; If ADC14DIF = 1: Ain+ = A16, Ain- = A17</p> <p>10010b = If ADC14DIF = 0: A18; If ADC14DIF = 1: Ain+ = A18, Ain- = A19</p> <p>10011b = If ADC14DIF = 0: A19; If ADC14DIF = 1: Ain+ = A18, Ain- = A19</p> <p>10100b = If ADC14DIF = 0: A20; If ADC14DIF = 1: Ain+ = A20, Ain- = A21</p> <p>10101b = If ADC14DIF = 0: A21; If ADC14DIF = 1: Ain+ = A20, Ain- = A21</p> <p>10110b = If ADC14DIF = 0: A22; If ADC14DIF = 1: Ain+ = A22, Ain- = A23</p> <p>10111b = If ADC14DIF = 0: A23; If ADC14DIF = 1: Ain+ = A22, Ain- = A23</p> <p>11000b = If ADC14DIF = 0: A24; If ADC14DIF = 1: Ain+ = A24, Ain- = A25</p> <p>11001b = If ADC14DIF = 0: A25; If ADC14DIF = 1: Ain+ = A24, Ain- = A25</p> <p>11010b = If ADC14DIF = 0: A26; If ADC14DIF = 1: Ain+ = A26, Ain- = A27</p> <p>11011b = If ADC14DIF = 0: A27; If ADC14DIF = 1: Ain+ = A26, Ain- = A27</p> <p>11100b = If ADC14DIF = 0: A28; If ADC14DIF = 1: Ain+ = A28, Ain- = A29</p> <p>11101b = If ADC14DIF = 0: A29; If ADC14DIF = 1: Ain+ = A28, Ain- = A29</p> <p>11110b = If ADC14DIF = 0: A30; If ADC14DIF = 1: Ain+ = A30, Ain- = A31</p> <p>11111b = If ADC14DIF = 0: A31; If ADC14DIF = 1: Ain+ = A30, Ain- = A31</p>

**20.3.8 ADC14MEM0 to ADC14MEM31 Register (offset = 098h to 104h) [reset = Undefined]**

ADC14 Conversion Memory x Register (x = 0 to 31)

**Figure 20-19. ADC14MEM0 to ADC14MEM31 Register**

31	30	29	28	27	26	25	24
Reserved							
r	r	r	r	r	r	r	r
Reserved							
r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8
Conversion_Results							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
Conversion_Results							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 20-11. ADC14MEM0 to ADC14MEM31 Register Description**

Bit	Field	Type	Reset	Description
31-16	Reserved	R	Undefined	Reserved.
15-0	Conversion Results	RW	Undefined	If ADC14DF = 0, unsigned binary: The 14-bit conversion results are right aligned. Bit 13 is the MSB. Bits 15-14 are 0 in 14-bit mode, bits 15-12 are 0 in 12-bit mode, bits 15-10 are 0 in 10-bit mode, and bits 15-8 are 0 in 8-bit mode. If the user writes to the conversion memory registers, the results are corrupted. If ADC14DF = 1, 2s-complement format: The 14-bit conversion results are left aligned. Bit 15 is the MSB. Bits 1-0 are 0 in 14-bit mode, bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode. The data is stored in the right-justified format and is converted to the left-justified 2s-complement format during read back. If the user writes to the conversion memory registers, the results are corrupted. Reading this register clears the corresponding bit in ADC14IFG0.

**NOTE:** ADC14MEMx register read by debugger does not clear the corresponding interrupt flag in ADC14IFG0 register.

### 20.3.9 ADC14IER0 Register (offset = 13Ch) [reset = 00000000h]

ADC14 Interrupt Enable 0 Register

**Figure 20-20. ADC14IER0 Register**

31	30	29	28	27	26	25	24
ADC14IE31	ADC14IE30	ADC14IE29	ADC14IE28	ADC14IE27	ADC14IE26	ADC14IE25	ADC14IE24
rw-0							
23	22	21	20	19	18	17	16
ADC14IE23	ADC14IE22	ADC14IE21	ADC14IE20	ADC14IE19	ADC14IE18	ADC14IE17	ADC14IE16
rw-0							
15	14	13	12	11	10	9	8
ADC14IE15	ADC14IE14	ADC14IE13	ADC14IE12	ADC14IE11	ADC14IE10	ADC14IE9	ADC14IE8
rw-0							
7	6	5	4	3	2	1	0
ADC14IE7	ADC14IE6	ADC14IE5	ADC14IE4	ADC14IE3	ADC14IE2	ADC14IE1	ADC14IE0
rw-0							

**Table 20-12. ADC14IER0 Register Description**

Bit	Field	Type	Reset	Description
31	ADC14IE31	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG31 bit. 0b = Interrupt disabled 1b = Interrupt enabled
30	ADC14IE30	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG30 bit. 0b = Interrupt disabled 1b = Interrupt enabled
29	ADC14IE29	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG29 bit. 0b = Interrupt disabled 1b = Interrupt enabled
28	ADC14IE28	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG28 bit. 0b = Interrupt disabled 1b = Interrupt enabled
27	ADC14IE27	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG27 bit. 0b = Interrupt disabled 1b = Interrupt enabled
26	ADC14IE26	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG26 bit. 0b = Interrupt disabled 1b = Interrupt enabled
25	ADC14IE25	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG25 bit. 0b = Interrupt disabled 1b = Interrupt enabled
24	ADC14IE24	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG24 bit. 0b = Interrupt disabled 1b = Interrupt enabled

**Table 20-12. ADC14IER0 Register Description (continued)**

Bit	Field	Type	Reset	Description
23	ADC14IE23	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG23 bit. 0b = Interrupt disabled 1b = Interrupt enabled
22	ADC14IE22	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG22 bit. 0b = Interrupt disabled 1b = Interrupt enabled
21	ADC14IE21	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG21 bit. 0b = Interrupt disabled 1b = Interrupt enabled
20	ADC14IE20	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG20 bit. 0b = Interrupt disabled 1b = Interrupt enabled
19	ADC14IE19	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG19 bit. 0b = Interrupt disabled 1b = Interrupt enabled
18	ADC14IE18	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG18 bit. 0b = Interrupt disabled 1b = Interrupt enabled
17	ADC14IE17	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG17 bit. 0b = Interrupt disabled 1b = Interrupt enabled
16	ADC14IE16	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG16 bit. 0b = Interrupt disabled 1b = Interrupt enabled
15	ADC14IE15	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG15 bit. 0b = Interrupt disabled 1b = Interrupt enabled
14	ADC14IE14	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG14 bit. 0b = Interrupt disabled 1b = Interrupt enabled
13	ADC14IE13	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG13 bit. 0b = Interrupt disabled 1b = Interrupt enabled
12	ADC14IE12	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG12 bit. 0b = Interrupt disabled 1b = Interrupt enabled
11	ADC14IE11	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG11 bit. 0b = Interrupt disabled 1b = Interrupt enabled

**Table 20-12. ADC14IER0 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
10	ADC14IE10	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG10 bit. 0b = Interrupt disabled 1b = Interrupt enabled
9	ADC14IE9	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG9 bit. 0b = Interrupt disabled 1b = Interrupt enabled
8	ADC14IE8	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG8 bit. 0b = Interrupt disabled 1b = Interrupt enabled
7	ADC14IE7	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG7 bit. 0b = Interrupt disabled 1b = Interrupt enabled
6	ADC14IE6	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG6 bit. 0b = Interrupt disabled 1b = Interrupt enabled
5	ADC14IE5	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG5 bit. 0b = Interrupt disabled 1b = Interrupt enabled
4	ADC14IE4	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG4 bit. 0b = Interrupt disabled 1b = Interrupt enabled
3	ADC14IE3	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG3 bit. 0b = Interrupt disabled 1b = Interrupt enabled
2	ADC14IE2	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG2 bit. 0b = Interrupt disabled 1b = Interrupt enabled
1	ADC14IE1	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG1 bit. 0b = Interrupt disabled 1b = Interrupt enabled
0	ADC14IE0	RW	0h	Interrupt enable. Enables or disables the interrupt request for the ADC14IFG0 bit. 0b = Interrupt disabled 1b = Interrupt enabled

**20.3.10 ADC14IER1 Register (offset = 140h) [reset = 00000000h]**

ADC14 Interrupt Enable 1 Register

**Figure 20-21. ADC14IER1 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
7	6	5	4	3	2	1	0
Reserved	ADC14RDYIE	ADC14TOVIE	ADC14OVIE	ADC14HIIIE	ADC14LOIE	ADC14INIE	Reserved
r-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0

**Table 20-13. ADC14IER1 Register Description**

Bit	Field	Type	Reset	Description
31-7	Reserved	R	0h	Reserved. Always reads as 0.
6	ADC14RDYIE	RW	0h	ADC14 local buffered reference ready interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
5	ADC14TOVIE	RW	0h	ADC14 conversion-time-overflow interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
4	ADC14OVIE	RW	0h	ADC14MEMx overflow-interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
3	ADC14HIIIE	RW	0h	Interrupt enable for the exceeding the upper limit interrupt of the window comparator for ADC14MEMx result register. 0b = Interrupt disabled 1b = Interrupt enabled
2	ADC14LOIE	RW	0h	Interrupt enable for the falling short of the lower limit interrupt of the window comparator for the ADC14MEMx result register. 0b = Interrupt disabled 1b = Interrupt enabled
1	ADC14INIE	RW	0h	Interrupt enable for the ADC14MEMx result register being greater than the ADC14LO threshold and below the ADC14HI threshold. 0b = Interrupt disabled 1b = Interrupt enabled
0	Reserved	R	0h	Reserved. Always reads as 0.

### 20.3.11 ADC14IFGR0 Register (offset = 144h) [reset = 00000000h]

ADC14 Interrupt Flag 0 Register

**Figure 20-22. ADC14IFGR0 Register**

31	30	29	28	27	26	25	24
ADC14IFG31	ADC14IFG30	ADC14IFG29	ADC14IFG28	ADC14IFG27	ADC14IFG26	ADC14IFG25	ADC14IFG24
r-0							
23	22	21	20	19	18	17	16
ADC14IFG23	ADC14IFG22	ADC14IFG21	ADC14IFG20	ADC14IFG19	ADC14IFG18	ADC14IFG17	ADC14IFG16
r-0							
15	14	13	12	11	10	9	8
ADC14IFG15	ADC14IFG14	ADC14IFG13	ADC14IFG12	ADC14IFG11	ADC14IFG10	ADC14IFG9	ADC14IFG8
r-0							
7	6	5	4	3	2	1	0
ADC14IFG7	ADC14IFG6	ADC14IFG5	ADC14IFG4	ADC14IFG3	ADC14IFG2	ADC14IFG1	ADC14IFG0
r-0							

**Table 20-14. ADC14IFGR0 Register Description**

Bit	Field	Type	Reset	Description
31	ADC14IFG31	R	0h	ADC14MEM31 interrupt flag. This bit is set to 1 when ADC14MEM31 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM31 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
30	ADC14IFG30	R	0h	ADC14MEM30 interrupt flag. This bit is set to 1 when ADC14MEM30 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM30 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
29	ADC14IFG29	R	0h	ADC14MEM29 interrupt flag. This bit is set to 1 when ADC14MEM29 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM29 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
28	ADC14IFG28	R	0h	ADC14MEM28 interrupt flag. This bit is set to 1 when ADC14MEM28 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM28 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
27	ADC14IFG27	R	0h	ADC14MEM27 interrupt flag. This bit is set to 1 when ADC14MEM27 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM27 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
26	ADC14IFG26	R	0h	ADC14MEM26 interrupt flag. This bit is set to 1 when ADC14MEM26 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM26 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
25	ADC14IFG25	R	0h	ADC14MEM25 interrupt flag. This bit is set to 1 when ADC14MEM25 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM25 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending

**Table 20-14. ADC14IFGR0 Register Description (continued)**

Bit	Field	Type	Reset	Description
24	ADC14IFG24	R	0h	ADC14MEM24 interrupt flag. This bit is set to 1 when ADC14MEM24 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM24 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
23	ADC14IFG23	R	0h	ADC14MEM23 interrupt flag. This bit is set to 1 when ADC14MEM23 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM23 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
22	ADC14IFG22	R	0h	ADC14MEM22 interrupt flag. This bit is set to 1 when ADC14MEM22 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM22 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
21	ADC14IFG21	R	0h	ADC14MEM21 interrupt flag. This bit is set to 1 when ADC14MEM21 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM21 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
20	ADC14IFG20	R	0h	ADC14MEM20 interrupt flag. This bit is set to 1 when ADC14MEM20 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM20 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
19	ADC14IFG19	R	0h	ADC14MEM19 interrupt flag. This bit is set to 1 when ADC14MEM19 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM19 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
18	ADC14IFG18	R	0h	ADC14MEM18 interrupt flag. This bit is set to 1 when ADC14MEM18 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM18 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
17	ADC14IFG17	R	0h	ADC14MEM17 interrupt flag. This bit is set to 1 when ADC14MEM17 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM17 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
16	ADC14IFG16	R	0h	ADC14MEM16 interrupt flag. This bit is set to 1 when ADC14MEM16 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM16 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
15	ADC14IFG15	R	0h	ADC14MEM15 interrupt flag. This bit is set to 1 when ADC14MEM15 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM15 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
14	ADC14IFG14	R	0h	ADC14MEM14 interrupt flag. This bit is set to 1 when ADC14MEM14 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM14 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending

**Table 20-14. ADC14IFGR0 Register Description (continued)**

Bit	Field	Type	Reset	Description
13	ADC14IFG13	R	0h	ADC14MEM13 interrupt flag. This bit is set to 1 when ADC14MEM13 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM13 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
12	ADC14IFG12	R	0h	ADC14MEM12 interrupt flag. This bit is set to 1 when ADC14MEM12 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM12 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
11	ADC14IFG11	R	0h	ADC14MEM11 interrupt flag. This bit is set to 1 when ADC14MEM11 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM11 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
10	ADC14IFG10	R	0h	ADC14MEM10 interrupt flag. This bit is set to 1 when ADC14MEM10 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM10 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
9	ADC14IFG9	R	0h	ADC14MEM9 interrupt flag. This bit is set to 1 when ADC14MEM9 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM9 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
8	ADC14IFG8	R	0h	ADC14MEM8 interrupt flag. This bit is set to 1 when ADC14MEM8 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM8 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
7	ADC14IFG7	R	0h	ADC14MEM7 interrupt flag. This bit is set to 1 when ADC14MEM7 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM7 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1.. 0b = No interrupt pending 1b = Interrupt pending
6	ADC14IFG6	R	0h	ADC14MEM6 interrupt flag. This bit is set to 1 when ADC14MEM6 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM6 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
5	ADC14IFG5	R	0h	ADC14MEM5 interrupt flag. This bit is set to 1 when ADC14MEM5 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM5 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
4	ADC14IFG4	R	0h	ADC14MEM4 interrupt flag. This bit is set to 1 when ADC14MEM4 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM4 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
3	ADC14IFG3	R	0h	ADC14MEM3 interrupt flag. This bit is set to 1 when ADC14MEM3 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM3 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending

**Table 20-14. ADC14IFGR0 Register Description (continued)**

Bit	Field	Type	Reset	Description
2	ADC14IFG2	R	0h	ADC14MEM2 interrupt flag. This bit is set to 1 when ADC14MEM2 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM2 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
1	ADC14IFG1	R	0h	ADC14MEM1 interrupt flag. This bit is set to 1 when ADC14MEM1 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM1 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending
0	ADC14IFG0	R	0h	ADC14MEM0 interrupt flag. This bit is set to 1 when ADC14MEM0 is loaded with a conversion result. This bit is reset to 0 when the ADC14MEM0 register is read, or when the corresponding bit in the ADC14CLRIFGR0 register is set to 1. 0b = No interrupt pending 1b = Interrupt pending

**20.3.12 ADC14IFGR1 Register (offset = 148h) [reset = 00000000h]**

ADC14 Interrupt Flag 1 Register

**Figure 20-23. ADC14IFGR1 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved	ADC14RDYIFG	ADC14TOVIFG	ADC14OVIFG	ADC14HIIIFG	ADC14LOIFG	ADC14INIFG	Reserved
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

**Table 20-15. ADC14IFGR1 Register Description**

Bit	Field	Type	Reset	Description
31-7	Reserved	R	0h	Reserved. Always reads as 0.
6	ADC14RDYIFG	R	0h	ADC14 local buffered reference ready interrupt flag. This bit is reset to 0 by IV register read or when corresponding bit in ADC14CLRIFGR1 is set to 1. 0b = No interrupt pending 1b = Interrupt pending
5	ADC14TOVIFG	R	0h	ADC14 conversion time overflow interrupt flag. This bit is reset to 0 by IV register read or when corresponding bit in ADC14CLRIFGR1 is set to 1. 0b = No interrupt pending 1b = Interrupt pending
4	ADC14OVIFG	R	0h	ADC14MEMx overflow interrupt flag. This bit is reset to 0 by IV register read or when corresponding bit in ADC14CLRIFGR1 is set to 1. 0b = No interrupt pending 1b = Interrupt pending
3	ADC14HIIIFG	R	0h	Interrupt flag for exceeding the upper limit interrupt of the window comparator for ADC14MEMx result register. This bit is reset to 0 by IV register read or when corresponding bit in ADC14CLRIFGR1 is set to 1. 0b = No interrupt pending 1b = Interrupt pending
2	ADC14LOIFG	R	0h	Interrupt flag for falling short of the lower limit interrupt of the window comparator for the ADC14MEMx result register. This bit is reset to 0 by IV register read or when corresponding bit in ADC14CLRIFGR1 is set to 1. 0b = No interrupt pending 1b = Interrupt pending
1	ADC14INIFG	R	0h	Interrupt flag for the ADC14MEMx result register being greater than the ADC14LO threshold and below the ADC14HI threshold interrupt. This bit is reset to 0 by IV register read or when corresponding bit in ADC14CLRIFGR1 is set to 1. 0b = No interrupt pending 1b = Interrupt pending
0	Reserved	R	0h	Reserved. Always reads as 0.

**20.3.13 ADC14CLRIFGR0 Register (offset = 14Ch) [reset = 00000000h]**

ADC14 Clear Interrupt Flag 0 Register

**Figure 20-24. ADC14CLRIFGR0 Register**

31	30	29	28	27	26	25	24
CLRADC14IFG 31	CLRADC14IFG 30	CLRADC14IFG 29	CLRADC14IFG 28	CLRADC14IFG 27	CLRADC14IFG 26	CLRADC14IFG 25	CLRADC14IFG 24
w-0							
23	22	21	20	19	18	17	16
CLRADC14IFG 23	CLRADC14IFG 22	CLRADC14IFG 21	CLRADC14IFG 20	CLRADC14IFG 19	CLRADC14IFG 18	CLRADC14IFG 17	CLRADC14IFG 16
w-0							
15	14	13	12	11	10	9	8
CLRADC14IFG 15	CLRADC14IFG 14	CLRADC14IFG 13	CLRADC14IFG 12	CLRADC14IFG 11	CLRADC14IFG 10	CLRADC14IFG 9	CLRADC14IFG 8
w-0							
7	6	5	4	3	2	1	0
CLRADC14IFG 7	CLRADC14IFG 6	CLRADC14IFG 5	CLRADC14IFG 4	CLRADC14IFG 3	CLRADC14IFG 2	CLRADC14IFG 1	CLRADC14IFG 0
w-0							

**Table 20-16. ADC14CLRIFGR0 Register Description**

Bit	Field	Type	Reset	Description
31	CLRADC14IFG31	W	0h	clear ADC14IFG31 0b = no effect 1b = clear pending interrupt flag
30	CLRADC14IFG30	W	0h	clear ADC14IFG30 0b = no effect 1b = clear pending interrupt flag
29	CLRADC14IFG29	W	0h	clear ADC14IFG29 0b = no effect 1b = clear pending interrupt flag
28	CLRADC14IFG28	W	0h	clear ADC14IFG28 0b = no effect 1b = clear pending interrupt flag
27	CLRADC14IFG27	W	0h	clear ADC14IFG27 0b = no effect 1b = clear pending interrupt flag
26	CLRADC14IFG26	W	0h	clear ADC14IFG26 0b = no effect 1b = clear pending interrupt flag
25	CLRADC14IFG25	W	0h	clear ADC14IFG25 0b = no effect 1b = clear pending interrupt flag
24	CLRADC14IFG24	W	0h	clear ADC14IFG24 0b = no effect 1b = clear pending interrupt flag
23	CLRADC14IFG23	W	0h	clear ADC14IFG23 0b = no effect 1b = clear pending interrupt flag

**Table 20-16. ADC14CLRIFGR0 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
22	CLRADC14IFG22	W	0h	clear ADC14IFG22 0b = no effect 1b = clear pending interrupt flag
21	CLRADC14IFG21	W	0h	clear ADC14IFG21 0b = no effect 1b = clear pending interrupt flag
20	CLRADC14IFG20	W	0h	clear ADC14IFG20 0b = no effect 1b = clear pending interrupt flag
19	CLRADC14IFG19	W	0h	clear ADC14IFG19 0b = no effect 1b = clear pending interrupt flag
18	CLRADC14IFG18	W	0h	clear ADC14IFG18 0b = no effect 1b = clear pending interrupt flag
17	CLRADC14IFG17	W	0h	clear ADC14IFG17 0b = no effect 1b = clear pending interrupt flag
16	CLRADC14IFG16	W	0h	clear ADC14IFG16 0b = no effect 1b = clear pending interrupt flag
15	CLRADC14IFG15	W	0h	clear ADC14IFG15 0b = no effect 1b = clear pending interrupt flag
14	CLRADC14IFG14	W	0h	clear ADC14IFG14 0b = no effect 1b = clear pending interrupt flag
13	CLRADC14IFG13	W	0h	clear ADC14IFG13 0b = no effect 1b = clear pending interrupt flag
12	CLRADC14IFG12	W	0h	clear ADC14IFG12 0b = no effect 1b = clear pending interrupt flag
11	CLRADC14IFG11	W	0h	clear ADC14IFG11 0b = no effect 1b = clear pending interrupt flag
10	CLRADC14IFG10	W	0h	clear ADC14IFG10 0b = no effect 1b = clear pending interrupt flag
9	CLRADC14IFG9	W	0h	clear ADC14IFG9 0b = no effect 1b = clear pending interrupt flag
8	CLRADC14IFG8	W	0h	clear ADC14IFG8 0b = no effect 1b = clear pending interrupt flag
7	CLRADC14IFG7	W	0h	clear ADC14IFG7 0b = no effect 1b = clear pending interrupt flag

**Table 20-16. ADC14CLRIFGR0 Register Description (continued)**

Bit	Field	Type	Reset	Description
6	CLRADC14IFG6	W	0h	clear ADC14IFG6 0b = no effect 1b = clear pending interrupt flag
5	CLRADC14IFG5	W	0h	clear ADC14IFG5 0b = no effect 1b = clear pending interrupt flag
4	CLRADC14IFG4	W	0h	clear ADC14IFG4 0b = no effect 1b = clear pending interrupt flag
3	CLRADC14IFG3	W	0h	clear ADC14IFG3 0b = no effect 1b = clear pending interrupt flag
2	CLRADC14IFG2	W	0h	clear ADC14IFG2 0b = no effect 1b = clear pending interrupt flag
1	CLRADC14IFG1	W	0h	clear ADC14IFG1 0b = no effect 1b = clear pending interrupt flag
0	CLRADC14IFG0	W	0h	clear ADC14IFG0 0b = no effect 1b = clear pending interrupt flag

**20.3.14 ADC14CLRIFGR1 Register (offset = 150h) [reset = 00000000h]**

ADC14 Clear Interrupt Flag 1 Register

**Figure 20-25. ADC14CLRIFGR1 Register**

31	30	29	28	27	26	25	24
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved	CLRADC14RDYIFG	CLRADC14TOVIFG	CLRADC14OVIFG	CLRADC14HIIIFG	CLRADC14LOIFG	CLRADC14INIFG	Reserved
r-0	w-0	w-0	w-0	w-0	w-0	w-0	r-0

**Table 20-17. ADC14CLRIFGR1 Register Description**

Bit	Field	Type	Reset	Description
31-7	Reserved	R	0h	Reserved. Always reads as 0.
6	CLRADC14RDYIFG	W	0h	clear ADC14RDYIFG 0b = no effect 1b = clear pending interrupt flag
5	CLRADC14TOVIFG	W	0h	clear ADC14TOVIFG 0b = no effect 1b = clear pending interrupt flag
4	CLRADC14OVIFG	W	0h	clear ADC14OVIFG 0b = no effect 1b = clear pending interrupt flag
3	CLRADC14HIIIFG	W	0h	clear ADC14HIIIFG 0b = no effect 1b = clear pending interrupt flag
2	CLRADC14LOIFG	W	0h	clear ADC14LOIFG 0b = no effect 1b = clear pending interrupt flag
1	CLRADC14INIFG	W	0h	clear ADC14INIFG 0b = no effect 1b = clear pending interrupt flag
0	Reserved	R	0h	Reserved. Always reads as 0.

**20.3.15 ADC14IV Register (offset = 154h) [reset = 00000000h]**

ADC14 Interrupt Vector Register

**Figure 20-26. ADC14IV Register**

31	30	29	28	27	26	25	24
ADC14IVx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
ADC14IVx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
15	14	13	12	11	10	9	8
ADC14IVx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
ADC14IVx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 20-18. ADC14IV Register Description**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
31-0	ADC14IVx	RW	0h	<p>ADC14 interrupt vector value. This register value is an encoded value of the highest pending interrupt in ADC14. Writing to this register clears all pending interrupt flags.</p> <p>00h = No interrupt pending</p> <p>02h = Interrupt Source: ADC14MEMx overflow; Interrupt Flag: ADC14OVIFG; Interrupt Priority: Highest</p> <p>04h = Interrupt Source: Conversion time overflow; Interrupt Flag: ADC14TOVIFG</p> <p>06h = Interrupt Source: ADC14 window high interrupt flag; Interrupt Flag: ADC14HIIFG</p> <p>08h = Interrupt Source: ADC14 window low interrupt flag; Interrupt Flag: ADC14LOIFG</p> <p>0Ah = Interrupt Source: ADC14 in-window interrupt flag; Interrupt Flag: ADC14INIFG</p> <p>0Ch = Interrupt Source: ADC14MEM0 interrupt flag; Interrupt Flag: ADC14IFG0</p> <p>0Eh = Interrupt Source: ADC14MEM1 interrupt flag; Interrupt Flag: ADC14IFG1</p> <p>10h = Interrupt Source: ADC14MEM2 interrupt flag; Interrupt Flag: ADC14IFG2</p> <p>12h = Interrupt Source: ADC14MEM3 interrupt flag; Interrupt Flag: ADC14IFG3</p> <p>14h = Interrupt Source: ADC14MEM4 interrupt flag; Interrupt Flag: ADC14IFG4</p> <p>16h = Interrupt Source: ADC14MEM5 interrupt flag; Interrupt Flag: ADC14IFG5</p> <p>18h = Interrupt Source: ADC14MEM6 interrupt flag; Interrupt Flag: ADC14IFG6</p> <p>1Ah = Interrupt Source: ADC14MEM7 interrupt flag; Interrupt Flag: ADC14IFG7</p> <p>1Ch = Interrupt Source: ADC14MEM8 interrupt flag; Interrupt Flag: ADC14IFG8</p> <p>1Eh = Interrupt Source: ADC14MEM9 interrupt flag; Interrupt Flag: ADC14IFG9</p> <p>20h = Interrupt Source: ADC14MEM10 interrupt flag; Interrupt Flag: ADC14IFG10</p> <p>22h = Interrupt Source: ADC14MEM11 interrupt flag; Interrupt Flag: ADC14IFG11</p> <p>24h = Interrupt Source: ADC14MEM12 interrupt flag; Interrupt Flag: ADC14IFG12</p> <p>26h = Interrupt Source: ADC14MEM13 interrupt flag; Interrupt Flag: ADC14IFG13</p> <p>28h = Interrupt Source: ADC14MEM14 interrupt flag; Interrupt Flag: ADC14IFG14</p> <p>2Ah = Interrupt Source: ADC14MEM15 interrupt flag; Interrupt Flag: ADC14IFG15</p> <p>2Ch = Interrupt Source: ADC14MEM16 interrupt flag; Interrupt Flag: ADC14IFG16</p> <p>2Eh = Interrupt Source: ADC14MEM17 interrupt flag; Interrupt Flag: ADC14IFG17</p> <p>30h = Interrupt Source: ADC14MEM18 interrupt flag; Interrupt Flag: ADC14IFG18</p> <p>32h = Interrupt Source: ADC14MEM19 interrupt flag; Interrupt Flag: ADC14IFG19</p> <p>34h = Interrupt Source: ADC14MEM20 interrupt flag; Interrupt Flag: ADC14IFG20</p> <p>36h = Interrupt Source: ADC14MEM21 interrupt flag; Interrupt Flag: ADC14IFG21</p> <p>38h = Interrupt Source: ADC14MEM22 interrupt flag; Interrupt Flag: ADC14IFG22</p> <p>3Ah = Interrupt Source: ADC14MEM23 interrupt flag; Interrupt Flag: ADC14IFG23</p> <p>3Ch = Interrupt Source: ADC14MEM24 interrupt flag; Interrupt Flag: ADC14IFG24</p> <p>3Eh = Interrupt Source: ADC14MEM25 interrupt flag; Interrupt Flag: ADC14IFG25</p> <p>40h = Interrupt Source: ADC14MEM26 interrupt flag; Interrupt Flag: ADC14IFG26</p> <p>42h = Interrupt Source: ADC14MEM27 interrupt flag; Interrupt Flag: ADC14IFG27</p> <p>44h = Interrupt Source: ADC14MEM28 interrupt flag; Interrupt Flag: ADC14IFG28</p> <p>46h = Interrupt Source: ADC14MEM29 interrupt flag; Interrupt Flag: ADC14IFG29</p> <p>48h = Interrupt Source: ADC14MEM30 interrupt flag; Interrupt Flag: ADC14IFG30</p> <p>4Ah = Interrupt Source: ADC14MEM31 interrupt flag; Interrupt Flag: ADC14IFG31</p> <p>4Ch = Interrupt Source: ADC14RDYIFG interrupt flag; Interrupt Flag: ADC14RDYIFG; Priority: Lowest</p>

## **Comparator E Module (COMP\_E)**

Comparator\_E is an analog voltage comparator with general comparator functionality for up to 16 channels. This chapter describes the Comparator\_E.

Topic	Page
21.1 COMP_E Introduction.....	709
21.2 COMP_E Operation .....	710
21.3 COMP_E Registers .....	716

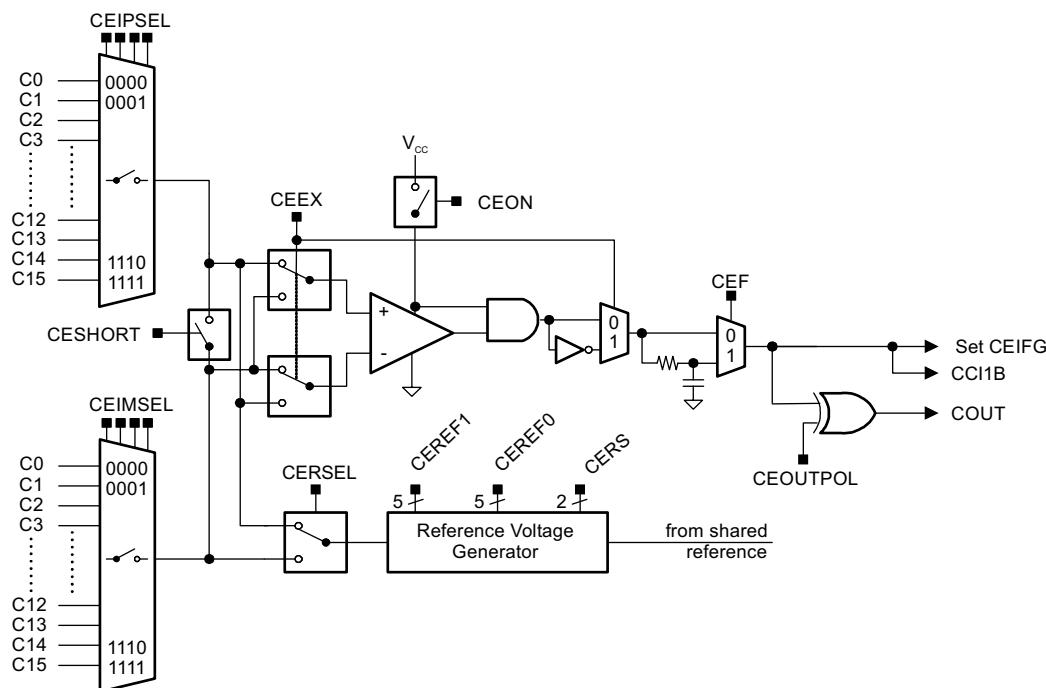
## 21.1 COMP\_E Introduction

The COMP\_E module supports precision slope analog-to-digital conversions, supply voltage supervision, and monitoring of external analog signals.

Features of COMP\_E include:

- Inverting and noninverting terminal input multiplexer
- Software-selectable RC filter for the comparator output
- Output provided to Timer\_A capture input
- Software control of the port input buffer
- Interrupt capability
- Selectable reference voltage generator, voltage hysteresis generator
- Reference voltage input from shared reference
- Ultra-low-power comparator mode
- Interrupt driven measurement system for low-power operation support

[Figure 21-1](#) shows the COMP\_E block diagram.



**Figure 21-1. COMP\_E Block Diagram**

## 21.2 COMP\_E Operation

The COMP\_E module is configured by user software. The setup and operation of COMP\_E is discussed in the following sections.

### 21.2.1 Comparator

The comparator compares the analog voltages at the positive (+) and negative (-) input terminals. If the + terminal is more positive than the – terminal, the comparator output CEOUT is high. The comparator can be switched on or off using control bit CEON. The comparator should be switched off when not in use to reduce current consumption. When the comparator is off, CEOUT is low when CEOUTPOL bit is set to 0, and CEOUT is high when CEOUTPOL bit is set to 1.

To optimize current consumption for the application, the lowest power mode that meets the comparator speed requirements (see the device-specific data sheet for the comparator propagation delay and response time) should be selected with the CEPWRMD bits. The CEPWRMD bits default to 0x0, which is the highest power and fastest speed. CEPWRMD = 0x2 is the lowest power and slowest speed option.

### 21.2.2 Analog Input Switches

The analog input switches connect or disconnect the two comparator input terminals to associated port pins using the CEIPSELx and CEIMSELx bits. The comparator terminal inputs can be controlled individually. The CEIPSELx and CEIMSELx bits allow:

- Application of an external signal to the V+ and V- terminals of the comparator
- Routing of an internal reference voltage to an associated output port pin
- Application of an external current source (for example, resistor) to the V+ or V- terminal of the comparator
- The mapping of both terminals of the internal multiplexer to the outside

Internally, the input switch is constructed as a T-switch to suppress distortion in the signal path.

---

**NOTE: Comparator Input Connection**

When the comparator is on, the input terminals should be connected to a signal, power, or ground. Otherwise, floating levels may cause unexpected interrupts and increased current consumption.

---

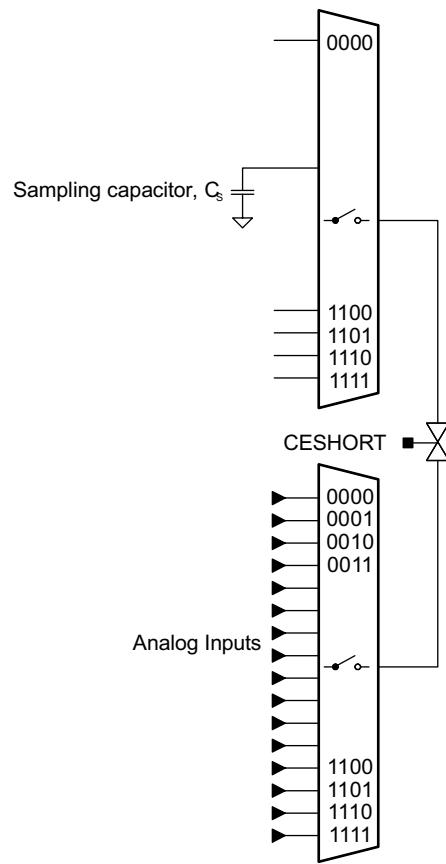
The CEEX bit controls the input multiplexer, permuting the input signals of the comparator V+ and V- terminals. Additionally, when the comparator terminals are permuted, the output signal from the comparator is inverted too. This allows the user to determine or compensate for the comparator input offset voltage.

### 21.2.3 Port Logic

The Px.y pins associated with a comparator channel are enabled by the CEIPSELx or CEIMSELx bits to disable its digital components while used as comparator input. Only one of the comparator input pins is selected as input to the comparator by the input multiplexer at a time.

### 21.2.4 Input Short Switch

The CESHORT bit shorts the Comparator\_E inputs. This can be used to build a simple sample-and-hold for the comparator (see [Figure 21-2](#)).



**Figure 21-2. COMP\_E Sample-And-Hold**

The required sampling time is proportional to the size of the sampling capacitor ( $C_s$ ), the resistance of the input switches in series with the short switch ( $R_i$ ), and the resistance of the external source ( $R_s$ ). The sampling capacitor  $C_s$  should be greater than 100 pF. The time constant, Tau, to charge the sampling capacitor  $C_s$  can be calculated with the following equation:

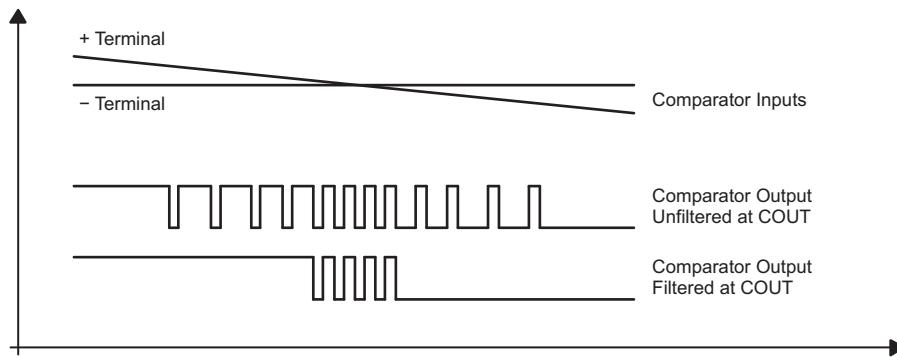
$$\text{Tau} = (R_i + R_s) \times C_s$$

Depending on the required accuracy, 3 to 10 Tau should be used as a sampling time. With 3 Tau the sampling capacitor is charged to approximately 95% of the input signals voltage level, with 5 Tau it is charged to more than 99%, and with 10 Tau the sampled voltage is sufficient for 12-bit accuracy.

### 21.2.5 Output Filter

The output of the comparator can be used with or without internal filtering. When control bit CEF is set, the output is filtered with an on-chip RC filter. The delay of the filter can be adjusted in four different steps.

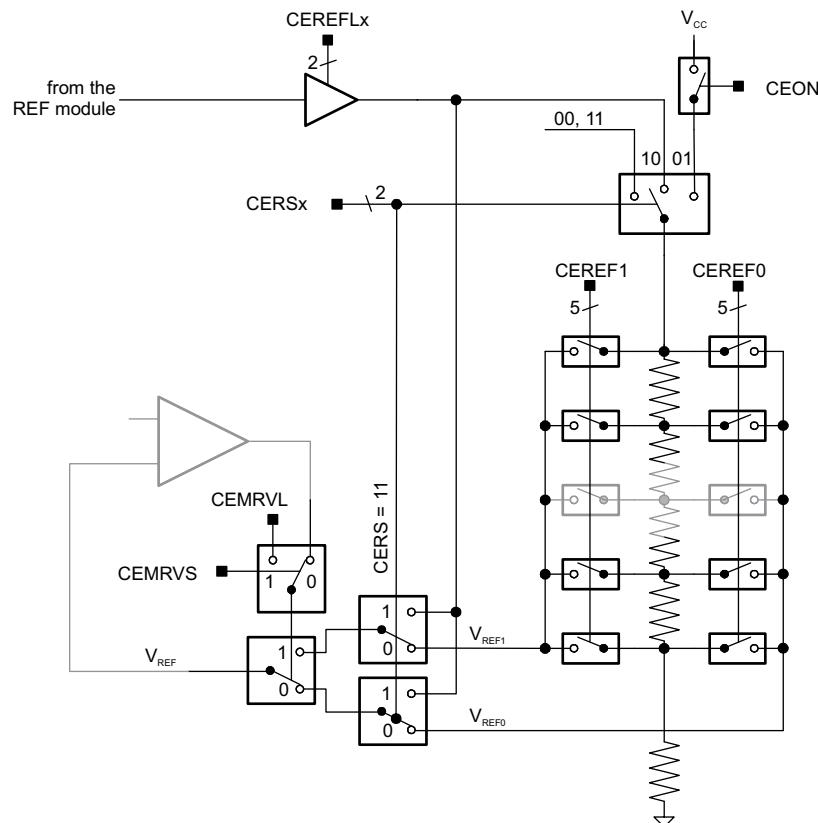
All comparator outputs oscillate if the voltage difference across the input terminals is small (see [Figure 21-3](#)). Internal and external parasitic effects and cross coupling on and between signal lines, power supply lines, and other parts of the system are responsible for this behavior. The comparator output oscillation reduces the accuracy and resolution of the comparison result. Selecting the output filter can reduce errors associated with comparator oscillation.



**Figure 21-3. RC-Filter Response at the Output of the Comparator**

### 21.2.6 Reference Voltage Generator

Figure 21-4 shows the Comparator\_E reference block diagram.



**Figure 21-4. Reference Generator Block Diagram**

The interrupt flags of the comparator and the comparator output are unchanged while the reference voltage from the shared reference is settling. If CERFLx is changed from a non-zero value to another non-zero value the interrupt flags may show unpredictable behavior. It is recommended to set CERFLx = 00 prior to changing the CERFLx settings.

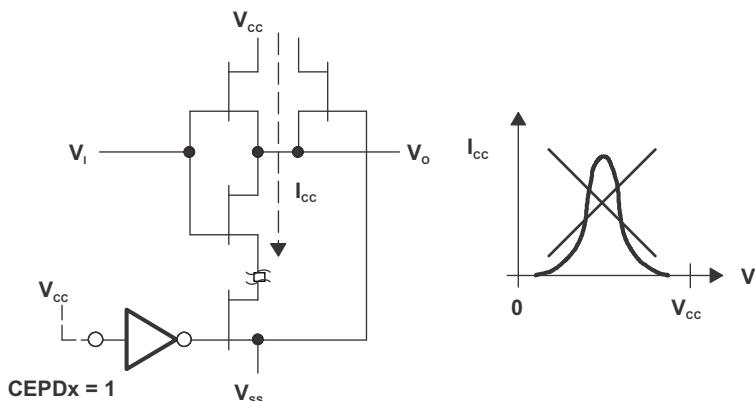
The voltage reference generator is used to generate VREF, which can be applied to either comparator input terminal. The CEREF1x (VREF1) and CEREF0x (VREF0) bits control the output of the voltage generator. The CERSEL bit selects the comparator terminal to which VREF is applied. If external signals are applied to both comparator input terminals, the internal reference generator should be turned off to reduce current consumption. The voltage reference generator can generate a fraction of the device's  $V_{CC}$  or of the voltage reference of the integrated precision voltage reference source. Vref1 is used while CEOUT is 1, and Vref0 is used while CEOUT is 0. This allows the generation of a hysteresis without using external components.

### 21.2.7 Port Disable Register (CEPD)

The comparator input and output functions are multiplexed with the associated I/O port pins, which are digital CMOS gates. When analog signals are applied to digital CMOS gates, parasitic current can flow from  $V_{CC}$  to GND. This parasitic current occurs if the input voltage is near the transition level of the gate. Disabling the port pin buffer eliminates the parasitic current flow and therefore reduces overall current consumption.

The CEPDx bits, when set, disable the corresponding Px.y input buffer as shown in Figure 21-5. When current consumption is critical, any Px.y pin connected to analog signals should be disabled with their associated CEPDx bits.

Selecting an input pin to the comparator multiplexer with the CEIPSEL or CEIMSEL bits automatically disables the input buffer for that pin, regardless of the state of the associated CEPDx bit.



**Figure 21-5. Transfer Characteristic and Power Dissipation in a CMOS Inverter/Buffer**

### 21.2.8 Comparator\_E Interrupts

One interrupt flag and one interrupt vector are associated with the Comparator\_E.

The interrupt flag CEIFG is set on either the rising or falling edge of the comparator output, selected by the CEIES bit. The comparator gives out an interrupt signal when both CEIFG and CEIE bits are set. The comparator interrupt can be serviced by the CPU when the comparator interrupt is enabled appropriately at the NVIC.

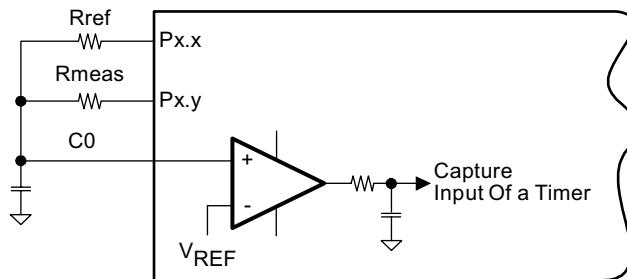
---

**NOTE:** Changing the value of CEIES bit might set the comparator interrupt flag CEIFG. This can happen even when the comparator is disabled (CEON = 0). It is recommended to clear CEIFG after configuring the comparator for proper interrupt behavior during operation.

---

### 21.2.9 Comparator\_E Used to Measure Resistive Elements

The Comparator\_E can be optimized to precisely measure resistive elements using single slope analog-to-digital conversion. For example, temperature can be converted into digital data using a thermistor, by comparing the thermistor's capacitor discharge time to that of a reference resistor (see Figure 21-6). A reference resistor  $R_{ref}$  is compared to  $R_{meas}$ .



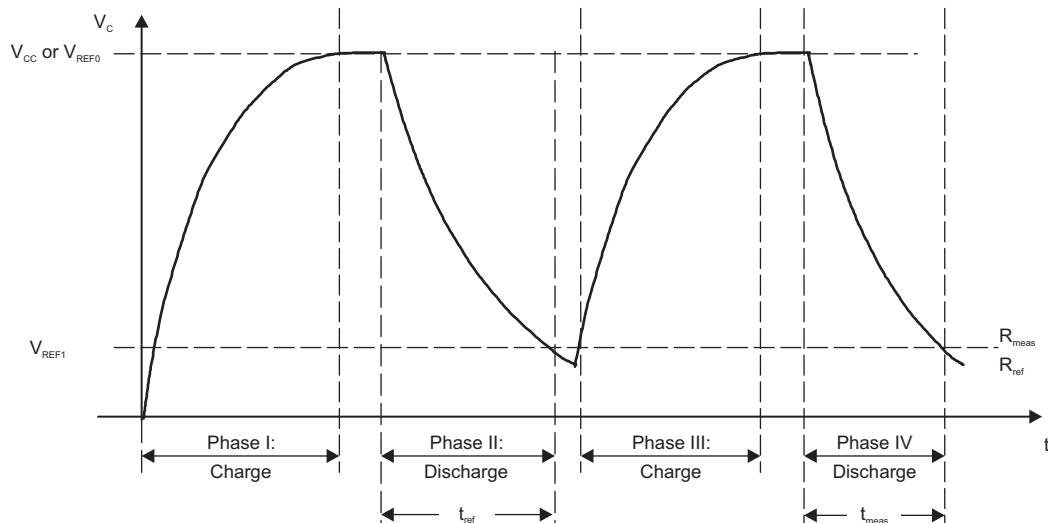
**Figure 21-6. Temperature Measurement System**

The resources used to calculate the temperature sensed by Rmeas are:

- Two digital I/O pins charge and discharge the capacitor.
- I/O is set to output high ( $V_{CC}$ ) to charge capacitor, reset to discharge.
- I/O is switched to high-impedance input with CEPDx set when not in use.
- One output charges and discharges the capacitor through Rref.
- One output discharges capacitor through Rmeas.
- The + terminal is connected to the positive terminal of the capacitor.
- The – terminal is connected to a reference level, for example  $0.25 \times V_{CC}$ .
- The output filter should be used to minimize switching noise.
- CEOUT is used to gate a timer capturing capacitor discharge time.

More than one resistive element can be measured. Additional elements are connected to C0 with available I/O pins and switched to high impedance when not being measured.

The thermistor measurement is based on a ratiometric conversion principle. The ratio of two capacitor discharge times is calculated as shown in [Figure 21-7](#).



**Figure 21-7. Timing for Temperature Measurement Systems**

The  $V_{CC}$  voltage and the capacitor value should remain constant during the conversion but are not critical, because they cancel in the ratio:

$$\frac{N_{meas}}{N_{ref}} = \frac{-R_{meas} \times C \times \ln\left(\frac{V_{ref1}}{V_{CC}}\right)}{-R_{ref} \times C \times \ln\left(\frac{V_{ref1}}{V_{CC}}\right)}$$

$$\frac{N_{meas}}{N_{ref}} = \frac{R_{meas}}{R_{ref}}$$

$$R_{meas} = R_{ref} \times \frac{N_{meas}}{N_{ref}}$$

## 21.3 COMP\_E Registers

[Table 21-1](#) lists the COMP\_E registers and their address offsets. Refer to the device-specific data sheet for the base address of the module.

**Table 21-1. COMP\_E Registers**

Offset	Acronym	Register	Type	Access	Reset	Section
00h	CExCTL0	Comparator_E control 0	Read/write	Half-word	0000h	<a href="#">Section 21.3.1</a>
02h	CExCTL1	Comparator_E control 1	Read/write	Half-word	0000h	<a href="#">Section 21.3.2</a>
04h	CExCTL2	Comparator_E control 2	Read/write	Half-word	0000h	<a href="#">Section 21.3.3</a>
06h	CExCTL3	Comparator_E control 3	Read/write	Half-word	0000h	<a href="#">Section 21.3.4</a>
0Ch	CExINT	Comparator_E interrupt	Read/write	Half-word	0000h	<a href="#">Section 21.3.5</a>
0Eh	CExIV	Comparator_E interrupt vector word	Read	Half-word	0000h	<a href="#">Section 21.3.6</a>

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

**21.3.1 CExCTL0 Register (offset = 00h) [reset = 0000h]**

Comparator\_E Control Register 0

**Figure 21-8. CExCTL0 Register**

15	14	13	12	11	10	9	8
CEIMEN	Reserved				CEIMSEL		
rw-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CEIPEN	Reserved				CEIPSEL		
rw-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0

**Table 21-2. CExCTL0 Register Description**

Bit	Field	Type	Reset	Description
15	CEIMEN	RW	0h	Channel input enable for the V- terminal of the comparator. 0b = Selected analog input channel for V- terminal is disabled. 1b = Selected analog input channel for V- terminal is enabled.
14-12	Reserved	R	0h	Reserved. Always reads as 0.
11-8	CEIMSEL	RW	0h	Channel input selected for the V- terminal of the comparator if CEIMEN is set to 1.
7	CEIPEN	RW	0h	Channel input enable for the V+ terminal of the comparator. 0b = Selected analog input channel for V+ terminal is disabled. 1b = Selected analog input channel for V+ terminal is enabled.
6-4	Reserved	R	0h	Reserved. Always reads as 0.
3-0	CEIPSEL	RW	0h	Channel input selected for the V+ terminal of the comparator if CEIPEN is set to 1.

**21.3.2 CExCTL1 Register (offset = 02h) [reset = 0000h]**

Comparator\_E Control Register 1

**Figure 21-9. CExCTL1 Register**

15	14	13	12	11	10	9	8
Reserved			CEMRVS	CEMRLV	CEON	CEPWRMD	
r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CEFDLY		CEEX	CESHORT	CEIES	CEF	CEOUTPOL	CEOUT
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0

**Table 21-3. CExCTL1 Register Description**

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved. Always reads as 0.
12	CEMRVS	RW	0h	This bit defines if the comparator output selects between VREF0 or VREF1 if CERS = 00, 01, or 10. 0b = Comparator output state selects between VREF0 or VREF1. 1b = CEMRVL selects between VREF0 or VREF1.
11	CEMRLV	RW	0h	This bit is valid if CEMRVS is set to 1. 0b = VREF0 is selected if CERS = 00, 01, or 10 1b = VREF1 is selected if CERS = 00, 01, or 10
10	CEON	RW	0h	On. This bit turns the comparator on. When the comparator is turned off the Comparator consumes no power. 0b = Off 1b = On
9-8	CEPWRMD	RW	0h	Power Mode 00b = High-speed mode 01b = Normal mode 10b = Ultra-low power mode 11b = Reserved
7-6	CEFDLY	RW	0h	Filter delay. The filter delay can be selected in four steps. See the device-specific data sheet for details. 00b = Typical filter delay of 500 ns 01b = Typical filter delay of 800 ns 10b = Typical filter delay of 1500 ns 11b = Typical filter delay of 3000 ns
5	CEEX	RW	0h	Exchange. This bit permutes the comparator 0 inputs and inverts the comparator 0 output.
4	CESHORT	RW	0h	Input short. This bit shorts the + and – input terminals. 0b = Inputs not shorted 1b = Inputs shorted
3	CEIES	RW	0h	Interrupt edge select for CEIIFG and CEIFG 0b = Rising edge for CEIFG, falling edge for CEIIFG 1b = Falling edge for CEIFG, rising edge for CEIIFG
2	CEF	RW	0h	Output filter. Available if CEPWRMD = 00, 01. 0b = Comparator output is not filtered 1b = Comparator output is filtered
1	CEOUTPOL	RW	0h	Output polarity. This bit defines the CEOUT polarity. 0b = Noninverted 1b = Inverted
0	CEOUT	RW	0h	Output value. This bit reflects the value of the Comparator output. Writing this bit has no effect on the comparator output.

### 21.3.3 CExCTL2 Register (offset = 04h) [reset = 0000h]

Comparator\_E Control Register 2

**Figure 21-10. CExCTL2 Register**

15	14	13	12	11	10	9	8
CEREFACC	CEREFL		CEREF1				
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
CERS	CERSEL		CEREOF0				
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 21-4. CExCTL2 Register Description**

Bit	Field	Type	Reset	Description
15	CEREFACC	RW	0h	Reference accuracy. A reference voltage is requested only if CEREFL > 0. 0b = Static mode 1b = Clocked (low power, low accuracy) mode
14-13	CEREFL	RW	0h	Reference voltage level 00b = Reference amplifier is disabled. No reference voltage is requested. 01b = 1.2 V is selected as shared reference voltage input 10b = 2.0 V is selected as shared reference voltage input 11b = 2.5 V is selected as shared reference voltage input
12-8	CEREF1	RW	0h	Reference resistor tap 1. This register defines the tap of the resistor string while CEOUT = 1.
7-6	CERS	RW	0h	Reference source. This bit defines if the reference voltage is derived from VCC or from the precise shared reference. 00b = No current is drawn by the reference circuitry. 01b = VCC applied to the resistor ladder 10b = Shared reference voltage applied to the resistor ladder. 11b = Shared reference voltage supplied to V(CREF). Resistor ladder is off.
5	CERSEL	RW	0h	Reference select. This bit selects which terminal the $V_{CREF}$ is applied to. 0b = When CEEX = 0, VREF is applied to the V+ terminal; When CEEX = 1, VREF is applied to the V- terminal 1b = When CEEX = 0, VREF is applied to the V- terminal; When CEEX = 1, VREF is applied to the V+ terminal
4-0	CEREOF0	RW	0h	Reference resistor tap 0. This register defines the tap of the resistor string while CEOUT = 0.

### 21.3.4 CExCTL3 Register (offset = 06h) [reset = 0000h]

Comparator\_E Control Register 3

**Figure 21-11. CExCTL3 Register**

15	14	13	12	11	10	9	8
CEPD15	CEPD14	CEPD13	CEPD12	CEPD11	CEPD10	CEPD9	CEPD8
rw-(0)							
7	6	5	4	3	2	1	0
CEPD7	CEPD6	CEPD5	CEPD4	CEPD3	CEPD2	CEPD1	CEPD0
rw-(0)							

**Table 21-5. CExCTL3 Register Description**

Bit	Field	Type	Reset	Description
15	CEPD15	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD15 disables the port of the comparator channel 15. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
14	CEPD14	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD14 disables the port of the comparator channel 14. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
13	CEPD13	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD13 disables the port of the comparator channel 13. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
12	CEPD12	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD12 disables the port of the comparator channel 12. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
11	CEPD11	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD11 disables the port of the comparator channel 11. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
10	CEPD10	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD10 disables the port of the comparator channel 10. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
9	CEPD9	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD9 disables the port of the comparator channel 9. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
8	CEPD8	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD8 disables the port of the comparator channel 8. 0b = The input buffer is enabled. 1b = The input buffer is disabled.

**Table 21-5. CExCTL3 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
7	CEPD7	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD7 disables the port of the comparator channel 7. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
6	CEPD6	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD6 disables the port of the comparator channel 6. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
5	CEPD5	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD5 disables the port of the comparator channel 5. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
4	CEPD4	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD4 disables the port of the comparator channel 4. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
3	CEPD3	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD3 disables the port of the comparator channel 3. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
2	CEPD2	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD2 disables the port of the comparator channel 2. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
1	CEPD1	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD1 disables the port of the comparator channel 1. 0b = The input buffer is enabled. 1b = The input buffer is disabled.
0	CEPD0	RW	0h	Port disable. These bits individually disable the input buffer for the pins of the port associated with Comparator_E. The bit CEPD0 disables the port of the comparator channel 0. 0b = The input buffer is enabled. 1b = The input buffer is disabled.

**21.3.5 CExINT Register (offset = 0Ch) [reset = 0000h]**

Comparator\_E Interrupt Control Register

**Figure 21-12. CExINT Register**

15	14	13	12	11	10	9	8
	Reserved		CERDYIE		Reserved	CEIIE	CEIE
r-0	r-0	r-0	rw-0	r-0	r-0	rw-0	rw-0
7	6	5	4	3	2	1	0
	Reserved		CERDYIFG		Reserved	CEIIFG	CEIFG
r-0	r-0	r-0	rw-0	r-0	r-0	rw-0	rw-0

**Table 21-6. CExINT Register Description**

Bit	Field	Type	Reset	Description
15-13	Reserved	R	0h	Reserved. Always reads as 0.
12	CERDYIE	RW	0h	Comparator ready interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
11-10	Reserved	R	0h	Reserved. Always reads as 0.
9	CEIIE	RW	0h	Comparator output interrupt enable inverted polarity 0b = Interrupt disabled 1b = Interrupt enabled
8	CEIE	RW	0h	Comparator output interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
7-5	Reserved	R	0h	Reserved. Always reads as 0.
4	CERDYIFG	RW	0h	Comparator ready interrupt flag. This bit is set if the Comparator reference sources are settled and the Comparator module is operational. This bit has to be cleared by software. 0b = No interrupt pending 1b = Interrupt pending
3-2	Reserved	R	0h	Reserved. Always reads as 0.
1	CEIIFG	RW	0h	Comparator output inverted interrupt flag. The bit CEIES defines the transition of the output setting this bit. 0b = No interrupt pending 1b = Interrupt pending
0	CEIFG	RW	0h	Comparator output interrupt flag. The bit CEIES defines the transition of the output setting this bit. 0b = No interrupt pending 1b = Interrupt pending

**21.3.6 CExIV Register (offset = 0Eh) [reset = 0000h]**

Comparator\_E Interrupt Vector Word Register

**Figure 21-13. CExIV Register**

15	14	13	12	11	10	9	8
CEIV							
r0	r0	r0	r0	r0	r0	r0	r0
CEIV							
r0	r0	r0	r0	r0	r-(0)	r-(0)	r0

**Table 21-7. CExIV Register Description**

Bit	Field	Type	Reset	Description
15-0	CEIV	R	0h	<p>Comparator interrupt vector word register. The interrupt vector register reflects only interrupt flags whose interrupt enable bit are set. Reading the CEIV register clears the pending interrupt flag with the highest priority.</p> <p>00h = No interrupt pending</p> <p>02h = Interrupt Source: CEOUT interrupt; Interrupt Flag: CEIFG; Interrupt Priority: Highest</p> <p>04h = Interrupt Source: CEOUT interrupt inverted polarity; Interrupt Flag: CEIIFG</p> <p>06h = Reserved</p> <p>08h = Reserved</p> <p>0Ah = Interrupt Source: Comparator ready interrupt; Interrupt Flag: CERDYIFG; Interrupt Priority: Lowest</p>

## ***Enhanced Universal Serial Communication Interface (eUSCI) – UART Mode***

---

---

The enhanced universal serial communication interface A (eUSCI\_A) supports multiple serial communication modes with one hardware module. This chapter discusses the operation of the asynchronous UART mode.

Topic	Page
22.1 Enhanced Universal Serial Communication Interface A (eUSCI_A) Overview .....	<a href="#">725</a>
22.2 eUSCI_A Introduction – UART Mode .....	<a href="#">725</a>
22.3 eUSCI_A Operation – UART Mode .....	<a href="#">727</a>
22.4 eUSCI_A UART Registers .....	<a href="#">743</a>

## 22.1 Enhanced Universal Serial Communication Interface A (eUSCI\_A) Overview

The eUSCI\_A module supports two serial communication modes:

- UART mode
- SPI mode

## 22.2 eUSCI\_A Introduction – UART Mode

In asynchronous mode, the eUSCI\_Ax modules connect the device to an external system through two external pins, UCAXRXD and UCAXTXD. UART mode is selected when the UCSYNC bit is cleared.

UART mode features include:

- 7-bit or 8-bit data with odd, even, or no parity
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- LSB-first or MSB-first data transmit and receive
- Built-in idle-line and address-bit communication protocols for multiprocessor systems
- Programmable baud rate with modulation for fractional baud-rate support
- Status flags for error detection and suppression
- Status flags for address detection
- Independent interrupt capability for receive, transmit, start bit received, and transmit complete

[Figure 22-1](#) shows the eUSCI\_Ax when configured for UART mode.

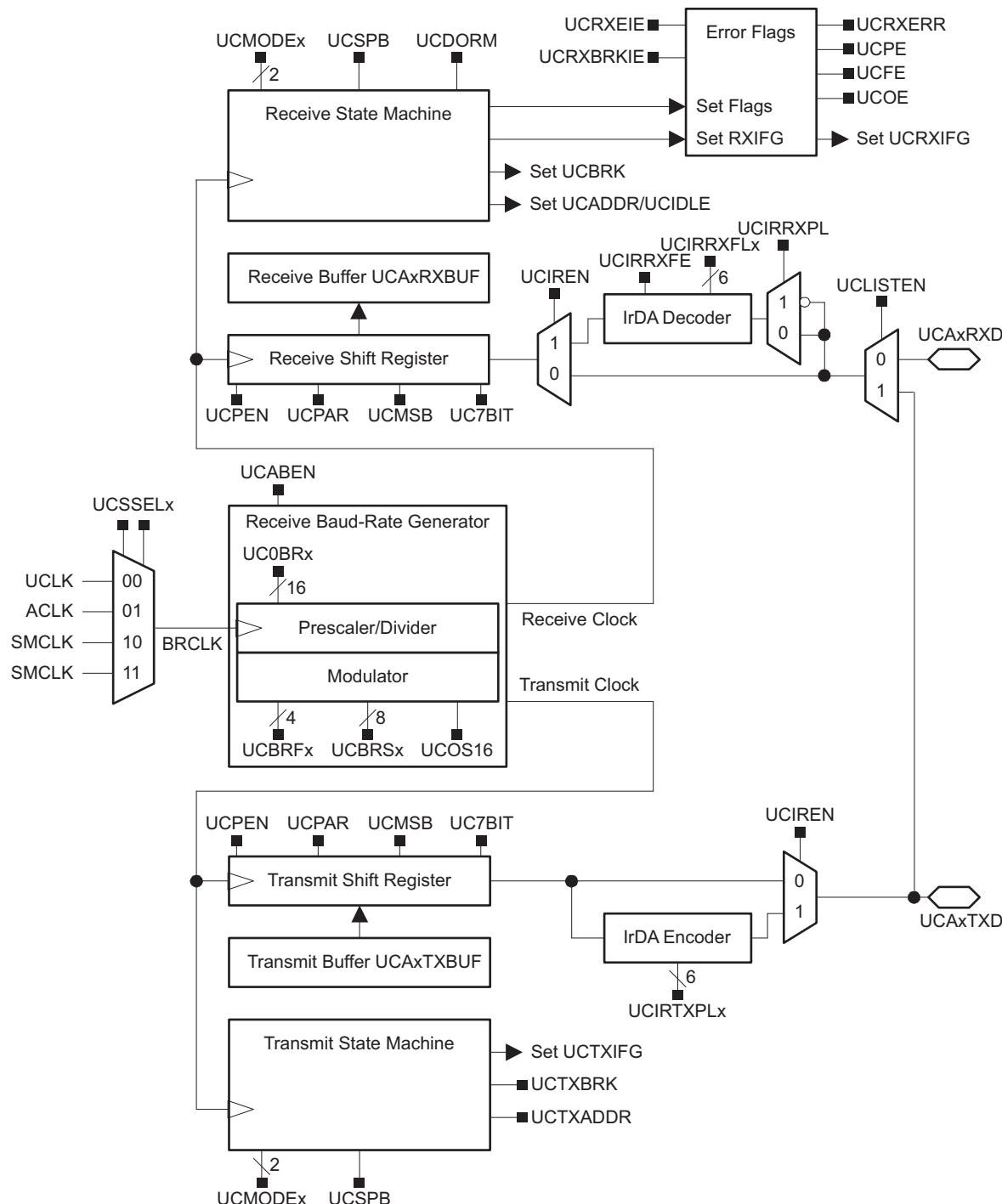


Figure 22-1. eUSCI\_Ax Block Diagram – UART Mode (UCSYNC = 0)

## 22.3 eUSCI\_A Operation – UART Mode

In UART mode, the eUSCI\_A transmits and receives characters at a bit rate asynchronous to another device. Timing for each character is based on the selected baud rate of the eUSCI\_A. The transmit and receive functions use the same baud-rate frequency.

### 22.3.1 eUSCI\_A Initialization and Reset

The eUSCI\_A is reset by a Hard Reset or by setting the UCSWRST bit. After a Hard Reset, the UCSWRST bit is automatically set, keeping the eUSCI\_A in a reset condition. When set, the UCSWRST bit sets the UCTXIFG bit and resets the UCRXIE, UCTXIE, UCRXIFG, UCRXERR, UCBRK, UCPE, UCOE, UCFE, UCSTOE, and UCBTOE bits. Clearing UCSWRST releases the eUSCI\_A for operation.

Configuring and reconfiguring the eUSCI\_A module should be done when UCSWRST is set to avoid unpredictable behavior.

---

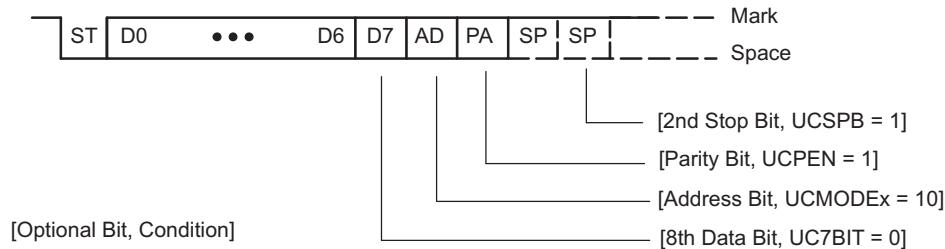
**NOTE: Initializing or reconfiguring the eUSCI\_A module**

The recommended eUSCI\_A initialization/reconfiguration process is:

1. Set UCSWRST.
  2. Initialize all eUSCI\_A registers with UCSWRST = 1 (including UCAXCTL1).
  3. Configure ports.
  4. Clear UCSWRST with software.
  5. Enable interrupts (optional) with UCRXIE or UCTXIE.
- 

### 22.3.2 Character Format

The UART character format (see [Figure 22-2](#)) consists of a start bit, seven or eight data bits, an even/odd/no parity bit, an address bit (address-bit mode), and one or two stop bits. The UCMSB bit controls the direction of the transfer and selects LSB or MSB first. LSB first is typically required for UART communication.



**Figure 22-2. Character Format**

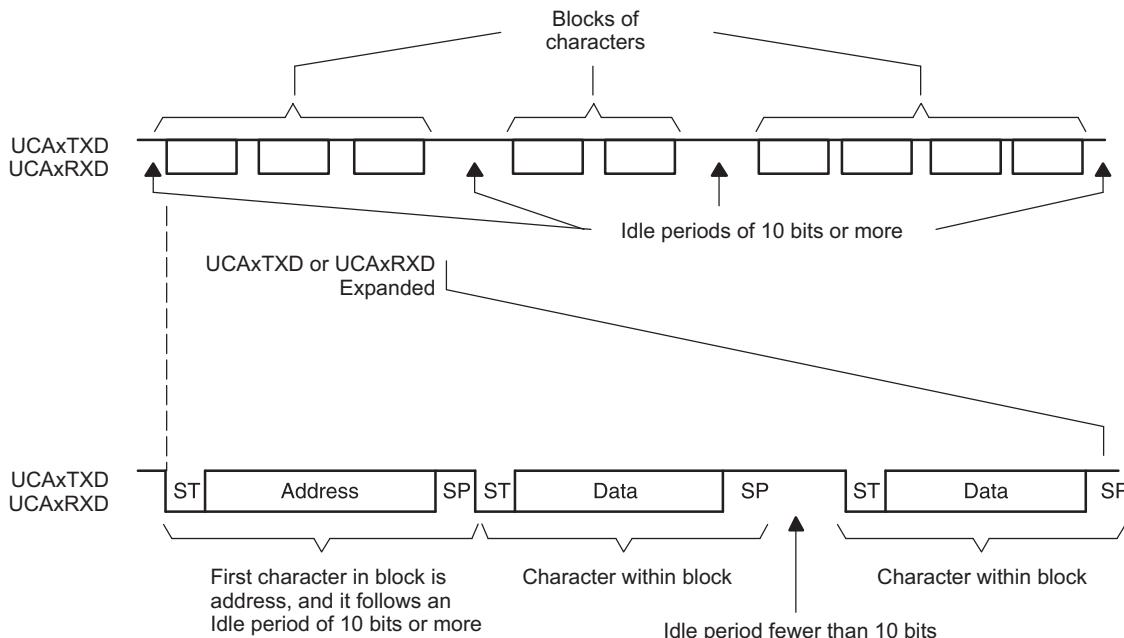
### 22.3.3 Asynchronous Communication Format

When two devices communicate asynchronously, no multiprocessor format is required for the protocol. When three or more devices communicate, the eUSCI\_A supports the idle-line and address-bit multiprocessor communication formats.

#### 22.3.3.1 Idle-Line Multiprocessor Format

When UCMODEx = 01, the idle-line multiprocessor format is selected. Blocks of data are separated by an idle time on the transmit or receive lines (see [Figure 22-3](#)). An idle receive line is detected when ten or more continuous ones (marks) are received after the one or two stop bits of a character. The baud-rate generator is switched off after reception of an idle line until the next start edge is detected. When an idle line is detected, the UCIDLE bit is set.

The first character received after an idle period is an address character. The UCIDLE bit is used as an address tag for each block of characters. In idle-line multiprocessor format, this bit is set when a received character is an address.



**Figure 22-3. Idle-Line Format**

The UCDORM bit is used to control data reception in the idle-line multiprocessor format. When UCDORM = 1, all non-address characters are assembled but not transferred into the UCAxRXBUF, and interrupts are not generated. When an address character is received, the character is transferred into UCAxRXBUF, UCRXIFG is set, and any applicable error flag is set when UCRXEIE = 1. When UCRXEIE = 0 and an address character is received but has a framing error or parity error, the character is not transferred into UCAxRXBUF and UCRXIFG is not set.

If an address is received, user software can validate the address and must reset UCDORM to continue receiving data. If UCDORM remains set, only address characters are received. When UCDORM is cleared during the reception of a character, the receive interrupt flag is set after the reception completed. The UCDORM bit is not modified automatically by the eUSCI\_A hardware.

For address transmission in idle-line multiprocessor format, a precise idle period can be generated by the eUSCI\_A to generate address character identifiers on UCAxTXD. The double-buffered UCTXADDR flag indicates if the next character loaded into UCAxTXBUF is preceded by an idle line of 11 bits. UCTXADDR is automatically cleared when the start bit is generated.

#### 22.3.3.1.1 Transmitting an Idle Frame

The following procedure sends out an idle frame to indicate an address character followed by associated data:

1. Set UCTXADDR, then write the address character to UCAxTXBUF. UCAxTXBUF must be ready for new data (UCTXIFG = 1).

This generates an idle period of exactly 11 bits followed by the address character. UCTXADDR is reset automatically when the address character is transferred from UCAxTXBUF into the shift register.

2. Write desired data characters to UCAxTXBUF. UCAxTXBUF must be ready for new data (UCTXIFG = 1).

The data written to UCAxTXBUF is transferred to the shift register and transmitted as soon as the shift register is ready for new data.

The idle-line time must not be exceeded between address and data transmission or between data transmissions. Otherwise, the transmitted data is misinterpreted as an address.

### 22.3.3.2 Address-Bit Multiprocessor Format

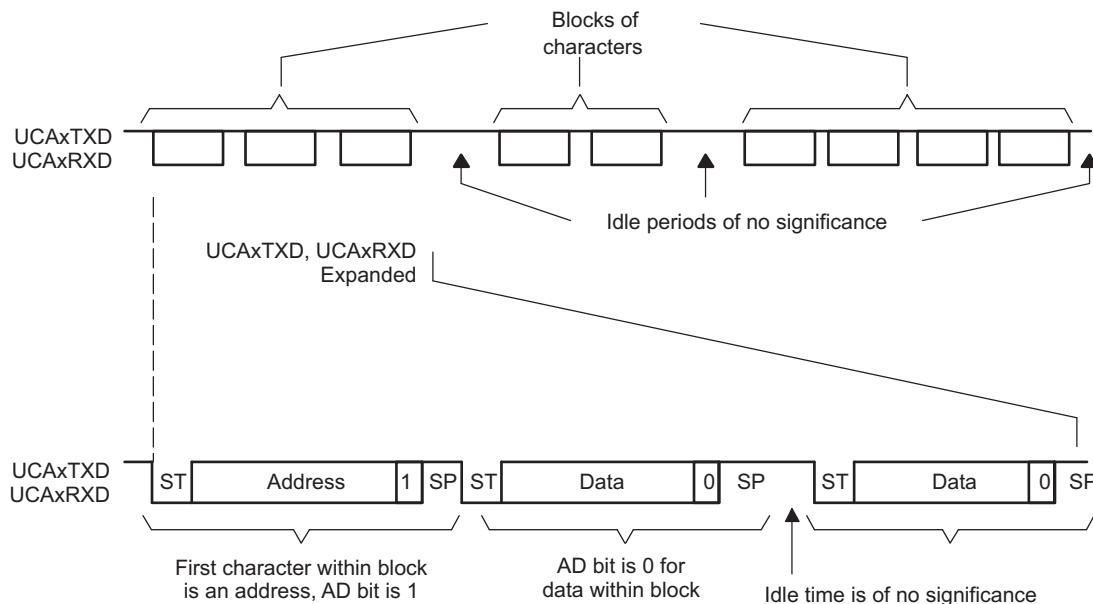
When UCMODEx = 10, the address-bit multiprocessor format is selected. Each processed character contains an extra bit used as an address indicator (see Figure 22-4). The first character in a block of characters carries a set address bit that indicates that the character is an address. The eUSCI\_A UCADDR bit is set when a received character has its address bit set and is transferred to UCAXRXBUF.

The UCDORM bit is used to control data reception in the address-bit multiprocessor format. When UCDORM is set, data characters with address bit = 0 are assembled by the receiver but are not transferred to UCAXRXBUF and no interrupts are generated. When a character containing a set address bit is received, the character is transferred into UCAXRXBUF, UCRXIFG is set, and any applicable error flag is set when UCRXEIE = 1. When UCRXEIE = 0 and a character containing a set address bit is received but has a framing error or parity error, the character is not transferred into UCAXRXBUF and UCRXIFG is not set.

If an address is received, user software can validate the address and must reset UCDORM to continue receiving data. If UCDORM remains set, only address characters with address bit = 1 are received. The UCDORM bit is not modified by the eUSCI\_A hardware automatically.

When UCDORM = 0, all received characters set the receive interrupt flag UCRXIFG. If UCDORM is cleared during the reception of a character, the receive interrupt flag is set after the reception is completed.

For address transmission in address-bit multiprocessor mode, the address bit of a character is controlled by the UCTXADDR bit. The value of the UCTXADDR bit is loaded into the address bit of the character transferred from UCAXTXBUF to the transmit shift register. UCTXADDR is automatically cleared when the start bit is generated.



**Figure 22-4. Address-Bit Multiprocessor Format**

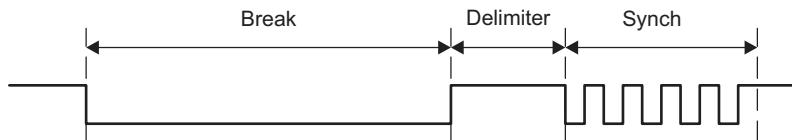
#### 22.3.3.2.1 Break Reception and Generation

When UCMODEx = 00, 01, or 10, the receiver detects a break when all data, parity, and stop bits are low, regardless of the parity, address mode, or other character settings. When a break is detected, the UCBRK bit is set. If the break interrupt enable bit (UCBRKIE) is set, the receive interrupt flag UCRXIFG is also set. In this case, the value in UCAXRXBUF is 0h, because all data bits were zero.

To transmit a break, set the UCTXBRK bit, then write 0h to UCAXTXBUF. UCAXTXBUF must be ready for new data (UCTXIFG = 1). This generates a break with all bits low. UCTXBRK is automatically cleared when the start bit is generated.

### 22.3.4 Automatic Baud-Rate Detection

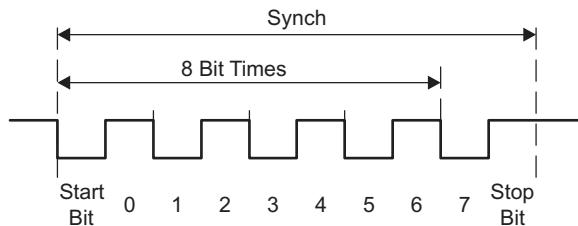
When UCMODEx = 11, UART mode with automatic baud-rate detection is selected. For automatic baud-rate detection, a data frame is preceded by a synchronization sequence that consists of a break and a synch field. A break is detected when 11 or more continuous zeros (spaces) are received. If the length of the break exceeds 21 bit times, the break timeout error flag UCBTOE is set. The eUSCI\_A cannot transmit data while receiving the break/sync field. The synch field follows the break as shown in [Figure 22-5](#).



**Figure 22-5. Auto Baud-Rate Detection – Break/Synch Sequence**

For LIN conformance, the character format should be set to eight data bits, LSB first, no parity, and one stop bit. No address bit is available.

The synch field consists of the data 055h inside a byte field (see [Figure 22-6](#)). The synchronization is based on the time measurement between the first falling edge and the last falling edge of the pattern. The transmit baud-rate generator is used for the measurement if automatic baud-rate detection is enabled by setting UCABDEN. Otherwise, the pattern is received but not measured. The result of the measurement is transferred into the baud-rate control registers (UCAxBRW and UCAxMCTLW). If the length of the synch field exceeds the measurable time, the synch timeout error flag UCSTOE is set. The result can be read after the receive interrupt flag UCRXIFG is set.



**Figure 22-6. Auto Baud-Rate Detection – Synch Field**

The UCDORM bit is used to control data reception in this mode. When UCDORM is set, all characters are received but not transferred into the UCAxRXBUF, and interrupts are not generated. When a break/synch field is detected, the UCBRK flag is set. The character following the break/synch field is transferred into UCAxRXBUF and the UCRXIFG interrupt flag is set. Any applicable error flag is also set. If the UCBRKIE bit is set, reception of the break/synch sets the UCRXIFG. The UCBRK bit is reset by user software or by reading the receive buffer UCAxRXBUF.

When a break/synch field is received, user software must reset UCDORM to continue receiving data. If UCDORM remains set, only the character after the next reception of a break/synch field is received. The UCDORM bit is not modified by the eUSCI\_A hardware automatically.

When UCDORM = 0, all received characters set the receive interrupt flag UCRXIFG. If UCDORM is cleared during the reception of a character, the receive interrupt flag is set after the reception is complete.

The counter used to detect the baud rate is limited to 0FFFFh ( $2^{16}$ ) counts. This means the minimum baud rate detectable is 244 baud in oversampling mode and 15 baud in low-frequency mode. The highest detectable baud rate is 1 Mbaud.

The automatic baud-rate detection mode can be used in a full-duplex communication system with some restrictions. The eUSCI\_A cannot transmit data while receiving the break/sync field and, if a 0h byte with framing error is received, any data transmitted during this time is corrupted. The latter case can be discovered by checking the received data and the UCFE bit.

### 22.3.4.1 Transmitting a Break/Synch Field

The following procedure transmits a break/synch field:

1. Set UCTXBRK with UMODEx = 11.
2. Write 055h to UCAXTXBUF. UCAXTXBUF must be ready for new data (UCTXIFG = 1).

This generates a break field of 13 bits followed by a break delimiter and the synch character. The length of the break delimiter is controlled with the UCDELIMx bits. UCTXBRK is reset automatically when the synch character is transferred from UCAXTXBUF into the shift register.

3. Write desired data characters to UCAXTXBUF. UCAXTXBUF must be ready for new data (UCTXIFG = 1).

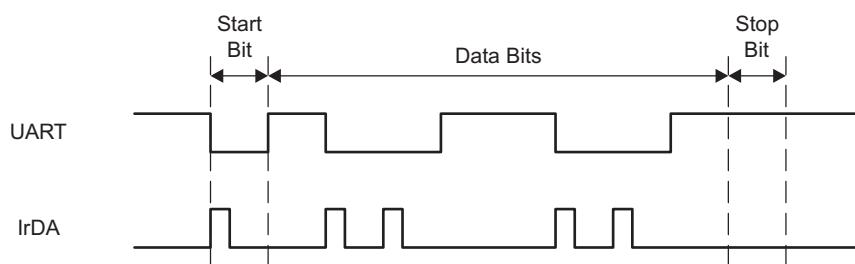
The data written to UCAXTXBUF is transferred to the shift register and transmitted as soon as the shift register is ready for new data.

### 22.3.5 IrDA Encoding and Decoding

When UCIREN is set, the IrDA encoder and decoder are enabled and provide hardware bit shaping for IrDA communication.

#### 22.3.5.1 IrDA Encoding

The encoder sends a pulse for every zero bit in the transmit bit stream coming from the UART (see Figure 22-7). The pulse duration is defined by UCIRTXPLx bits specifying the number of one-half clock periods of the clock selected by UCIRTXCLK.



**Figure 22-7. UART vs IrDA Data Format**

To set the pulse time of 3/16 bit period required by the IrDA standard, the BITCLK16 clock is selected with UCIRTXCLK = 1, and the pulse length is set to six one-half clock cycles with UCIRTXPLx = 6 – 1 = 5.

When UCIRTXCLK = 0, the pulse length  $t_{PULSE}$  is based on BRCLK and is calculated as:

$$\text{UCIRTXPLx} = t_{PULSE} \times 2 \times f_{BRCLK} - 1$$

When UCIRTXCLK = 0, the prescaler UCBRx must be set to a value greater or equal to 5.

#### 22.3.5.2 IrDA Decoding

The decoder detects high pulses when UCIRRXPL = 0. Otherwise, it detects low pulses. In addition to the analog deglitch filter, an additional programmable digital filter stage can be enabled by setting UCIRRXFE. When UCIRRXFE is set, only pulses longer than the programmed filter length are passed. Shorter pulses are discarded. The equation to program the filter length UCIRRXFLx is:

$$\text{UCIRRXFLx} = (t_{PULSE} - t_{WAKE}) \times 2 \times f_{BRCLK} - 4$$

Where:

$t_{PULSE}$  = Minimum receive pulse duration

$t_{WAKE}$  = Wake time from any low-power mode. Zero when the device is in active mode.

## 22.3.6 Automatic Error Detection

Glitch suppression prevents the eUSCI\_A from being accidentally started. Any pulse on UC $A_x$ RXD shorter than the deglitch time  $t_g$  (selected by UCGLITx) is ignored (see the device-specific data sheet for parameters).

When a low period on UC $A_x$ RXD exceeds  $t_g$ , a majority vote is taken for the start bit. If the majority vote fails to detect a valid start bit, the eUSCI\_A halts character reception and waits for the next low period on UC $A_x$ RXD. The majority vote is also used for each bit in a character to prevent bit errors.

The eUSCI\_A module automatically detects framing errors, parity errors, overrun errors, and break conditions when receiving characters. The bits UCFE, UCPE, UCOE, and UCBRK are set when their respective condition is detected. When the error flags UCFE, UCPE, or UCOE are set, UCRXERR is also set. [Table 22-1](#) describes the error conditions.

**Table 22-1. Receive Error Conditions**

Error Condition	Error Flag	Description
Framing error	UCFE	A framing error occurs when a low stop bit is detected. When two stop bits are used, both stop bits are checked for framing error. When a framing error is detected, the UCFE bit is set.
Parity error	UCPE	A parity error is a mismatch between the number of 1s in a character and the value of the parity bit. When an address bit is included in the character, it is included in the parity calculation. When a parity error is detected, the UCPE bit is set.
Receive overrun	UCOE	An overrun error occurs when a character is loaded into UC $A_x$ RXB $U$ before the prior character has been read. When an overrun occurs, the UCOE bit is set.
Break condition	UCBRK	When not using automatic baud-rate detection, a break is detected when all data, parity, and stop bits are low. When a break condition is detected, the UCBRK bit is set. A break condition can also set the interrupt flag UCRXIFG if the break interrupt enable UCBRKIE bit is set.

When UCRXEIE = 0 and a framing error or parity error is detected, no character is received into UC $A_x$ RXB $U$ . When UCRXEIE = 1, characters are received into UC $A_x$ RXB $U$  and any applicable error bit is set.

When any of the UCFE, UCPE, UCOE, UCBRK, or UCRXERR bit is set, the bit remains set until user software resets it or UC $A_x$ RXB $U$  is read. UCOE must be reset by reading UC $A_x$ RXB $U$ . Otherwise, it does not function properly. To detect overflows reliably, the following flow is recommended. After a character is received and UC $A_x$ RXIFG is set, first read UC $A_x$ STATW to check the error flags including the overflow flag UCOE. Read UC $A_x$ RXB $U$  next. This clears all error flags except UCOE, if UC $A_x$ RXB $U$  was overwritten between the read access to UC $A_x$ STATW and to UC $A_x$ RXB $U$ . Therefore, the UCOE flag should be checked after reading UC $A_x$ RXB $U$  to detect this condition. Note that, in this case, the UCRXERR flag is not set.

### 22.3.7 eUSCI\_A Receive Enable

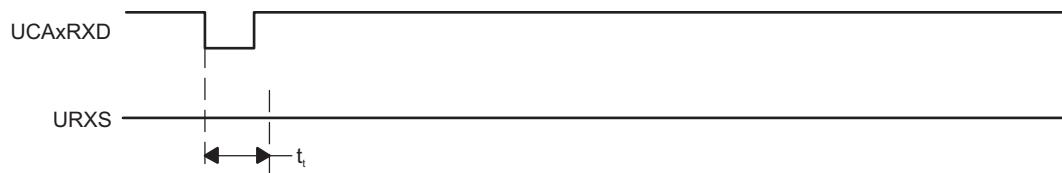
The eUSCI\_A module is enabled by clearing the UCSWRST bit and the receiver is ready and in an idle state. The receive baud rate generator is in a ready state but is not clocked nor producing any clocks.

The falling edge of the start bit enables the baud rate generator and the UART state machine checks for a valid start bit. If no valid start bit is detected the UART state machine returns to its idle state and the baud rate generator is turned off again. If a valid start bit is detected, a character is received.

When the idle-line multiprocessor mode is selected with UCMODEx = 01, the UART state machine checks for an idle line after receiving a character. If a start bit is detected, another character is received. Otherwise, the UCIDLE flag is set after 10 ones are received, the UART state machine returns to its idle state, and the baud rate generator is turned off.

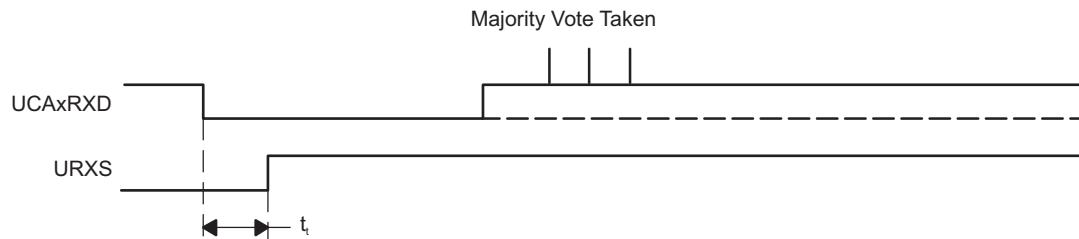
#### 22.3.7.1 Receive Data Glitch Suppression

Glitch suppression prevents the eUSCI\_A from being accidentally started. Any glitch on UCAXRXD shorter than the deglitch time  $t_g$  is ignored by the eUSCI\_A, and further action is initiated as shown in [Figure 22-8](#) (see the device-specific data sheet for parameters). The deglitch time  $t_g$  can be set to four different values using the UCGLITx bits.



**Figure 22-8. Glitch Suppression, eUSCI\_A Receive Not Started**

When a glitch is longer than  $t_g$ , or a valid start bit occurs on UCAXRXD, the eUSCI\_A receive operation is started and a majority vote is taken (see [Figure 22-9](#)). If the majority vote fails to detect a start bit, the eUSCI\_A halts character reception.



**Figure 22-9. Glitch Suppression, eUSCI\_A Activated**

### 22.3.8 eUSCI\_A Transmit Enable

The eUSCI\_A module is enabled by clearing the UCSWRST bit and the transmitter is ready and in an idle state. The transmit baud-rate generator is ready but is not clocked nor producing any clocks.

A transmission is initiated by writing data to UCAXTXBUF. When this occurs, the baud-rate generator is enabled, and the data in UCAXTXBUF is moved to the transmit shift register on the next BITCLK after the transmit shift register is empty. UCTXIFG is set when new data can be written into UCAXTXBUF.

Transmission continues as long as new data is available in UCAXTXBUF at the end of the previous byte transmission. If new data is not in UCAXTXBUF when the previous byte has transmitted, the transmitter returns to its idle state and the baud-rate generator is turned off.

## 22.3.9 UART Baud-Rate Generation

The eUSCI\_A baud-rate generator is capable of producing standard baud rates from nonstandard source frequencies. It provides two modes of operation selected by the UCOS16 bit.

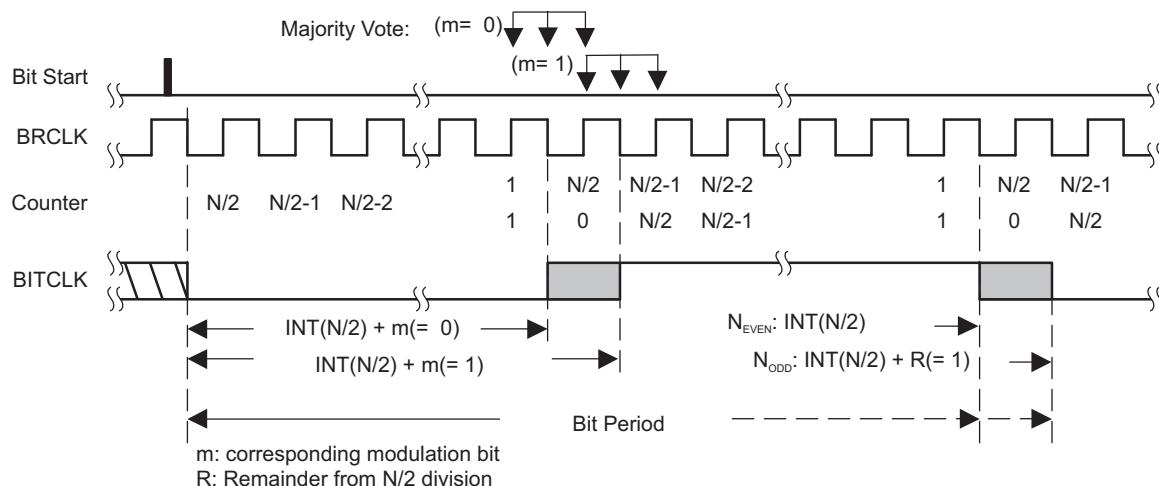
A quick setup for finding the correct baud-rate settings for the eUSCI\_A can be found in [Section 22.3.10](#).

### 22.3.9.1 Low-Frequency Baud-Rate Generation

The low-frequency mode is selected when UCOS16 = 0. This mode allows generation of baud rates from low-frequency clock sources (for example, 9600 baud from a 32768-Hz crystal). By using a lower input frequency, the power consumption of the module is reduced. Using this mode with higher frequencies and higher prescaler settings causes the majority votes to be taken in an increasingly smaller window and, thus, decrease the benefit of the majority vote.

In low-frequency mode, the baud-rate generator uses one prescaler and one modulator to generate bit clock timing. This combination supports fractional divisors for baud-rate generation. In this mode, the maximum eUSCI\_A baud rate is one-third the UART source clock frequency BRCLK.

Timing for each bit is shown in [Figure 22-10](#). For each bit received, a majority vote is taken to determine the bit value. These samples occur at the  $N/2 - 1/2$ ,  $N/2$ , and  $N/2 + 1/2$  BRCLK periods, where N is the number of BRCLK cycles per BITCLK.



**Figure 22-10. BITCLK Baud-Rate Timing With UCOS16 = 0**

Modulation is based on the UCBRSx setting as shown in [Table 22-2](#). A 1 in the table indicates that  $m = 1$  and the corresponding BITCLK period is one BRCLK period longer than a BITCLK period with  $m = 0$ . The modulation wraps around after 8 bits but restarts with each new start bit.

**Table 22-2. Modulation Pattern Examples**

UCBRSx	Bit 0 (Start Bit)	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
0x00	0	0	0	0	0	0	0	0
0x01	0	0	0	0	0	0	0	1
⋮								
0x35	0	0	1	1	0	1	0	1
0x36	0	0	1	1	0	1	1	0
0x37	0	0	1	1	0	1	1	1
⋮								
0xFF	1	1	1	1	1	1	1	1

[Section 22.3.10](#) describes the correct setting of UCBRSx.

### 22.3.9.2 Oversampling Baud-Rate Generation

The oversampling mode is selected when UCOS16 = 1. This mode supports sampling a UART bit stream with higher input clock frequencies. This results in majority votes that are always 1/16 of a bit clock period apart. This mode also easily supports IrDA pulses with a 3/16 bit time when the IrDA encoder and decoder are enabled.

This mode uses one prescaler and one modulator to generate the BITCLK16 clock that is 16 times faster than the BITCLK. An additional divider by 16 and modulator stage generates BITCLK from BITCLK16. This combination supports fractional divisions of both BITCLK16 and BITCLK for baud-rate generation. In this mode, the maximum eUSCI\_A baud rate is 1/16 the UART source clock frequency BRCLK.

Modulation for BITCLK16 is based on the UCBRFx setting (see [Table 22-3](#)). A 1 in the table indicates that the corresponding BITCLK16 period is one BRCLK period longer than the periods  $m = 0$ . The modulation restarts with each new bit timing.

Modulation for BITCLK is based on the UCBRSp setting as previously described.

**Table 22-3. BITCLK16 Modulation Pattern**

UCBRFx	Number of BITCLK16 Clocks After Last Falling BITCLK Edge															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
03h	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
04h	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1
05h	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1
06h	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1
07h	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1
08h	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1
09h	0	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1
0Ah	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1
0Bh	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1
0Ch	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1
0Dh	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1
0Eh	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0Fh	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

### 22.3.10 Setting a Baud Rate

For a given BRCLK clock source, the baud rate used determines the required division factor N:

$$N = f_{BRCLK} / \text{baud rate}$$

The division factor N is often a noninteger value, thus, at least one divider and one modulator stage is used to meet the factor as closely as possible.

If N is equal or greater than 16, it is recommended to use the oversampling baud-rate generation mode by setting UCOS16.

---

**NOTE: Baud-rate settings quick set up**

To calculate the correct settings for the baud-rate generation, perform these steps:

1. Calculate  $N = f_{BRCLK} / \text{baud rate}$  [if  $N > 16$  continue with step 3, otherwise with step 2]
  2.  $\text{OS16} = 0$ ,  $\text{UCBRx} = \text{INT}(N)$  [continue with step 4]
  3.  $\text{OS16} = 1$ ,  $\text{UCBRx} = \text{INT}(N/16)$ ,  $\text{UCBRFx} = \text{INT}([(N/16) - \text{INT}(N/16)] \times 16)$
  4.  $\text{UCBRSx}$  can be found by looking up the fractional part of N ( $= N - \text{INT}(N)$ ) in table [Table 22-4](#)
  5. If  $\text{OS16} = 0$  was chosen, a detailed error calculation is recommended to be performed
- 

[Table 22-4](#) can be used as a lookup table for finding the correct UCBRSx modulation pattern for the corresponding fractional part of N. The values there are optimized for transmitting.

**Table 22-4. UCBRSx Settings for Fractional Portion of  $N = f_{BRCLK}/\text{Baud Rate}$**

Fractional Portion of N	UCBRSx <sup>(1)</sup>	Fractional Portion of N	UCBRSx <sup>(1)</sup>
0.0000	0x00	0.5002	0xAA
0.0529	0x01	0.5715	0x6B
0.0715	0x02	0.6003	0xAD
0.0835	0x04	0.6254	0xB5
0.1001	0x08	0.6432	0xB6
0.1252	0x10	0.6667	0xD6
0.1430	0x20	0.7001	0xB7
0.1670	0x11	0.7147	0xBB
0.2147	0x21	0.7503	0xDD
0.2224	0x22	0.7861	0xED
0.2503	0x44	0.8004	0xEE
0.3000	0x25	0.8333	0xBF
0.3335	0x49	0.8464	0xDF
0.3575	0x4A	0.8572	0xEF
0.3753	0x52	0.8751	0xF7
0.4003	0x92	0.9004	0xFB
0.4286	0x53	0.9170	0xFD
0.4378	0x55	0.9288	0xFE

<sup>(1)</sup> The UCBRSx setting in one row is valid from the fractional portion given in that row until the one in the next row

#### 22.3.10.1 Low-Frequency Baud-Rate Mode Setting

In low-frequency mode, the integer portion of the divisor is realized by the prescaler:

$$\text{UCBRx} = \text{INT}(N)$$

The fractional portion is realized by the modulator with its UCBRSx setting. The recommended way of determining the correct UCBRSx is performing a detailed error calculation as explained in the following sections. However it is also possible to look up the correct settings in table with typical crystals (see [Table 22-5](#)).

### 22.3.10.2 Oversampling Baud-Rate Mode Setting

In the oversampling mode, the prescaler is set to:

$$\text{UCBRx} = \text{INT}(N / 16)$$

and the first stage modulator is set to:

$$\text{UCBRFx} = \text{INT}([(N / 16) - \text{INT}(N / 16)] \times 16)$$

The second modulation stage setting (UCBRSx) can be found by performing a detailed error calculation or by using [Table 22-4](#) and the fractional part of  $N = f_{\text{BRCLK}} / \text{baud rate}$ .

### 22.3.11 Transmit Bit Timing - Error calculation

The timing for each character is the sum of the individual bit timings. Using the modulation features of the baud-rate generator reduces the cumulative bit error. The individual bit error can be calculated using the following steps.

#### 22.3.11.1 Low-Frequency Baud-Rate Mode Bit Timing

In low-frequency mode, calculation of the length of bit  $i T_{\text{bit,TX}}[i]$  is based on the UCBRx and UCBRSx settings:

$$T_{\text{bit,TX}}[i] = (1/f_{\text{BRCLK}})(\text{UCBRx} + m_{\text{UCBRSx}}[i])$$

Where:

$m_{\text{UCBRSx}}[i]$  = Modulation of bit  $i$  of UCBRSx

#### 22.3.11.2 Oversampling Baud-Rate Mode Bit Timing

In oversampling baud-rate mode, calculation of the length of bit  $i T_{\text{bit,TX}}[i]$  is based on the baud-rate generator UCBRx, UCBRFx and UCBRSx settings:

$$t_{\text{bit,TX}}[i] = \frac{1}{f_{\text{BRCLK}}} \left( (16 \times \text{UCBRx}) + \sum_{j=0}^{15} m_{\text{UCBRFx}}[j] + m_{\text{UCBRSx}}[i] \right)$$

Where:

$\sum_{j=0}^{15} m_{\text{UCBRFx}}[j]$  = Sum of ones from the corresponding row in [Table 22-3](#)

$m_{\text{UCBRSx}}[i]$  = Modulation of bit  $i$  of UCBRSx

This results in an end-of-bit time  $t_{\text{bit,TX}}[i]$  equal to the sum of all previous and the current bit times:

$$t_{\text{bit,TX}}[i] = \sum_{j=0}^i t_{\text{bit,TX}}[j]$$

To calculate bit error, this time is compared to the ideal bit time  $t_{\text{bit,ideal,TX}}[i]$ :

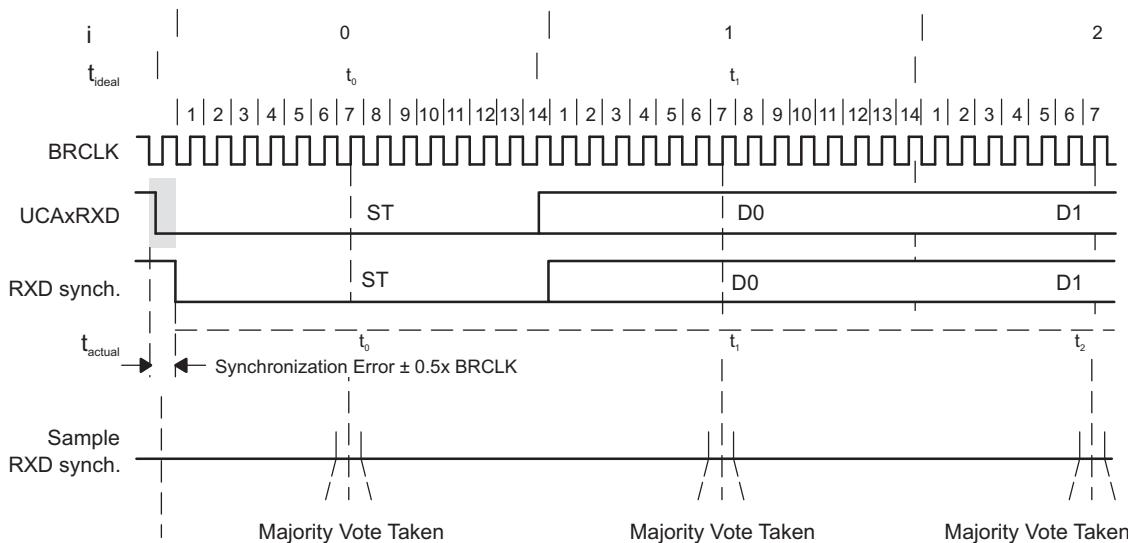
$$t_{\text{bit,ideal,TX}}[i] = (1 / \text{baud rate})(i + 1)$$

This results in an error normalized to one ideal bit time ( $1 / \text{baud rate}$ ):

$$\text{Error}_{\text{TX}}[i] = (t_{\text{bit,TX}}[i] - t_{\text{bit,ideal,TX}}[i]) \times \text{baud rate} \times 100\%$$

### 22.3.12 Receive Bit Timing – Error Calculation

Receive timing error consists of two error sources. The first is the bit-to-bit timing error similar to the transmit bit timing error. The second is the error between a start edge occurring and the start edge being accepted by the eUSCI\_A module. [Figure 22-11](#) shows the asynchronous timing errors between data on the UCAXRXD pin and the internal baud-rate clock. This results in an additional synchronization error. The synchronization error  $t_{\text{SYNC}}$  is between  $-0.5$  BRCLK cycles and  $+0.5$  BRCLK cycles, independent of the selected baud-rate generation mode.



**Figure 22-11. Receive Error**

The ideal sampling time  $t_{bit,ideal,RX}[i]$  is in the middle of a bit period:

$$t_{bit,ideal,RX}[i] = (1 / \text{baud rate})(i + 0.5)$$

The real sampling time,  $t_{bit,RX}[i]$ , is equal to the sum of all previous bits according to the formulas shown in the transmit timing section, plus one-half BITCLK for the current bit i, plus the synchronization error  $t_{SYNC}$ .

This results in the following  $t_{bit,RX}[i]$  for the low-frequency baud-rate mode:

$$t_{bit,RX}[i] = t_{SYNC} + \sum_{j=0}^{i-1} T_{bit,RX}[j] + \frac{1}{f_{BRCLK}} \left( \text{INT}(1/2UCBRx) + m_{UCBRSx}[i] \right)$$

Where:

$$T_{bit,RX}[i] = (1/f_{BRCLK})(UCBRx + m_{UCBRSx}[i])$$

$m_{UCBRSx}[i]$  = Modulation of bit i of UCBRSx

For the oversampling baud-rate mode, the sampling time  $t_{bit,RX}[i]$  of bit i is calculated by:

$$t_{bit,RX}[i] = t_{SYNC} + \sum_{j=0}^{i-1} T_{bit,RX}[j] + \frac{1}{f_{BRCLK}} \left( (8 * UCBRx) + \sum_{j=0}^7 m_{UCBFRx}[j] + m_{UCBRSx}[i] \right)$$

Where:

$$T_{bit,RX}[i] = \frac{1}{f_{BRCLK}} \left( (16 \times UCBRx) + \sum_{j=0}^{15} m_{UCBFRx}[j] + m_{UCBRSx}[i] \right)$$

$$\sum_{j=0}^{7+m_{UCBRSx}[i]} m_{UCBFRx}[j]$$

= Sum of ones from columns 0 to  $(7 + m_{UCBRSx}[i])$  from the corresponding row in

**Table 22-3.**

$m_{UCBRSx}[i]$  = Modulation of bit i of UCBRSx

This results in an error normalized to one ideal bit time ( $1 / \text{baud rate}$ ) according to the following formula:

$$\text{Error}_{RX}[i] = (t_{bit,RX}[i] - t_{bit,ideal,RX}[i]) \times \text{baud rate} \times 100\%$$

### 22.3.13 Typical Baud Rates and Errors

Standard baud-rate data for UCBRx, UCBRSx, and UCBRFx are listed in [Table 22-5](#) for a 32768-Hz crystal sourcing ACLK and typical SMCLK frequencies. Make sure that the selected BRCLK frequency does not exceed the device specific maximum eUSCI\_A input frequency (see the device-specific data sheet).

The receive error is the accumulated time versus the ideal scanning time in the middle of each bit. The worst-case error is given for the reception of an 8-bit character with parity and one stop bit including synchronization error.

The transmit error is the accumulated timing error versus the ideal time of the bit period. The worst-case error is given for the transmission of an 8-bit character with parity and stop bit.

**Table 22-5. Recommended Settings for Typical Crystals and Baud Rates<sup>(1)</sup>**

BRCLK	Baud Rate	UCOS16	UCBRx	UCBRFx	UCBRSx	TX Error <sup>(2)</sup> (%)		RX Error <sup>(2)</sup> (%)	
						neg	pos	neg	pos
32768	1200	1	1	11	0x25	-2.29	2.25	-2.56	5.35
32768	2400	0	13	–	0xB6	-3.12	3.91	-5.52	8.84
32768	4800	0	6	–	0xEE	-7.62	8.98	-21	10.25
32768	9600	0	3	–	0x92	-17.19	16.02	-23.24	37.3
1000000	9600	1	6	8	0x20	-0.48	0.64	-1.04	1.04
1000000	19200	1	3	4	0x2	-0.8	0.96	-1.84	1.84
1000000	38400	1	1	10	0x0	0	1.76	0	3.44
1000000	57600	0	17	–	0x4A	-2.72	2.56	-3.76	7.28
1000000	115200	0	8	–	0xD6	-7.36	5.6	-17.04	6.96
1048576	9600	1	6	13	0x22	-0.46	0.42	-0.48	1.23
1048576	19200	1	3	6	0xAD	-0.88	0.83	-2.36	1.18
1048576	38400	1	1	11	0x25	-2.29	2.25	-2.56	5.35
1048576	57600	0	18	–	0x11	-2	3.37	-5.31	5.55
1048576	115200	0	9	–	0x08	-5.37	4.49	-5.93	14.92
4000000	9600	1	26	0	0xB6	-0.08	0.16	-0.28	0.2
4000000	19200	1	13	0	0x84	-0.32	0.32	-0.64	0.48
4000000	38400	1	6	8	0x20	-0.48	0.64	-1.04	1.04
4000000	57600	1	4	5	0x55	-0.8	0.64	-1.12	1.76
4000000	115200	1	2	2	0xBB	-1.44	1.28	-3.92	1.68
4000000	230400	0	17	–	0x4A	-2.72	2.56	-3.76	7.28
4194304	9600	1	27	4	0xFB	-0.11	0.1	-0.33	0
4194304	19200	1	13	10	0x55	-0.21	0.21	-0.55	0.33
4194304	38400	1	6	13	0x22	-0.46	0.42	-0.48	1.23
4194304	57600	1	4	8	0xEE	-0.75	0.74	-2	0.87
4194304	115200	1	2	4	0x92	-1.62	1.37	-3.56	2.06
4194304	230400	0	18	–	0x11	-2	3.37	-5.31	5.55
8000000	9600	1	52	1	0x49	-0.08	0.04	-0.1	0.14
8000000	19200	1	26	0	0xB6	-0.08	0.16	-0.28	0.2
8000000	38400	1	13	0	0x84	-0.32	0.32	-0.64	0.48
8000000	57600	1	8	10	0xF7	-0.32	0.32	-1	0.36
8000000	115200	1	4	5	0x55	-0.8	0.64	-1.12	1.76
8000000	230400	1	2	2	0xBB	-1.44	1.28	-3.92	1.68
8000000	460800	0	17	–	0x4A	-2.72	2.56	-3.76	7.28
8388608	9600	1	54	9	0xEE	-0.06	0.06	-0.11	0.13
8388608	19200	1	27	4	0xFB	-0.11	0.1	-0.33	0
8388608	38400	1	13	10	0x55	-0.21	0.21	-0.55	0.33
8388608	57600	1	9	1	0xB5	-0.31	0.31	-0.53	0.78
8388608	115200	1	4	8	0xEE	-0.75	0.74	-2	0.87

<sup>(1)</sup> The listed UCBRSx settings are determined by a search algorithm for the lowest error. Other settings for UCBRSx might result in similar or same errors.

<sup>(2)</sup> Assumes a stable clock source for BRCLK with negligible jitter (for example, from a crystal oscillator). Any frequency variation or jitter of the clock source will make the errors worse.

**Table 22-5. Recommended Settings for Typical Crystals and Baud Rates<sup>(1)</sup> (continued)**

BRCLK	Baud Rate	UCOS16	UCBRx	UCBRFx	UCBRSx	TX Error <sup>(2)</sup> (%)		RX Error <sup>(2)</sup> (%)	
						neg	pos	neg	pos
8388608	230400	1	2	4	0x92	-1.62	1.37	-3.56	2.06
8388608	460800	0	18	–	0x11	-2	3.37	-5.31	5.55
12000000	9600	1	78	2	0x0	0	0	0	0.04
12000000	19200	1	39	1	0x0	0	0	0	0.16
12000000	38400	1	19	8	0x65	-0.16	0.16	-0.4	0.24
12000000	57600	1	13	0	0x25	-0.16	0.32	-0.48	0.48
12000000	115200	1	6	8	0x20	-0.48	0.64	-1.04	1.04
12000000	230400	1	3	4	0x2	-0.8	0.96	-1.84	1.84
12000000	460800	1	1	10	0x0	0	1.76	0	3.44
16000000	9600	1	104	2	0xD6	-0.04	0.02	-0.09	0.03
16000000	19200	1	52	1	0x49	-0.08	0.04	-0.1	0.14
16000000	38400	1	26	0	0xB6	-0.08	0.16	-0.28	0.2
16000000	57600	1	17	5	0xDD	-0.16	0.2	-0.3	0.38
16000000	115200	1	8	10	0xF7	-0.32	0.32	-1	0.36
16000000	230400	1	4	5	0x55	-0.8	0.64	-1.12	1.76
16000000	460800	1	2	2	0xBB	-1.44	1.28	-3.92	1.68
16777216	9600	1	109	3	0xB5	-0.03	0.02	-0.05	0.06
16777216	19200	1	54	9	0xEE	-0.06	0.06	-0.11	0.13
16777216	38400	1	27	4	0xFB	-0.11	0.1	-0.33	0
16777216	57600	1	18	3	0x44	-0.16	0.15	-0.2	0.45
16777216	115200	1	9	1	0xB5	-0.31	0.31	-0.53	0.78
16777216	230400	1	4	8	0xEE	-0.75	0.74	-2	0.87
16777216	460800	1	2	4	0x92	-1.62	1.37	-3.56	2.06
20000000	9600	1	130	3	0x25	-0.02	0.03	0	0.07
20000000	19200	1	65	1	0xD6	-0.06	0.03	-0.1	0.1
20000000	38400	1	32	8	0xEE	-0.1	0.13	-0.27	0.14
20000000	57600	1	21	11	0x22	-0.16	0.13	-0.16	0.38
20000000	115200	1	10	13	0xAD	-0.29	0.26	-0.46	0.66
20000000	230400	1	5	6	0xEE	-0.67	0.51	-1.71	0.62
20000000	460800	1	2	11	0x92	-1.38	0.99	-1.84	2.8

### 22.3.14 Using the eUSCI\_A Module in UART Mode With Low-Power Modes

The eUSCI module is not functional when the device is in LPM3, LPM4, or LPMx.5 modes of operation. However, the application can make use of the FORCE\_LPM\_ENTRY bit in the PCMCTL1 register to determine whether the entry to low-power mode should be aborted if the eUSCI is active, or if the device can continue low power entry regardless. The latter option is useful if the eUSCI is transmitting or receiving data at a very slow rate, and the application should enter low-power mode at the expense of a packet of data being lost. Refer to the *Power Control Manager (PCM)* chapter for more details.

### 22.3.15 eUSCI\_A Interrupts

The eUSCI\_A has only one interrupt vector that is shared for transmission and for reception.

#### 22.3.15.1 eUSCI\_A Transmit Interrupt Operation

The UCTXIFG interrupt flag is set by the transmitter to indicate that UCAXTXBUF is ready to accept another character. An interrupt request is generated if UCTXIE is set. UCTXIFG is automatically reset if a character is written to UCAXTXBUF.

UCTXIFG is set after a Hard Reset or when UCSWRST = 1. UCTXIE is reset after a Hard Reset or when UCSWRST = 1.

### 22.3.15.2 eUSCI\_A Receive Interrupt Operation

The UCRXIFG interrupt flag is set each time a character is received and loaded into UC $Ax$ RXB $UF$ . An interrupt request is generated if UCRXIE is set. UCRXIFG and UCRXIE are reset by a Hard Reset signal or when UCSWRST = 1. UCRXIFG is automatically reset when UC $Ax$ RXB $UF$  is read.

Additional interrupt control features include:

- When UC $Ax$ RXEIE = 0, erroneous characters do not set UCRXIFG.
- When UCDORM = 1, nonaddress characters do not set UCRXIFG in multiprocessor modes. In plain UART mode, no characters set UCRXIFG.
- When UCBRKIE = 1, a break condition sets the UCBRK bit and the UCRXIFG flag.

### 22.3.15.3 eUSCI\_A Receive Interrupt Operation

[Table 22-6](#) describes the UART state change interrupt flags.

**Table 22-6. UART State Change Interrupt Flags**

Interrupt Flag	Interrupt Condition
UCSTTIFG	START byte received interrupt. This flag is set when the UART module receives a START byte.
UCTXCPTIFG	Transmit complete interrupt. This flag is set, after the complete UART byte in the internal shift register including STOP bit got shifted out and UC $Ax$ TXB $UF$ is empty.

### 22.3.15.4 UC $Ax$ IV, Interrupt Vector Generator

The eUSCI\_A interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register UC $Ax$ IV is used to determine which flag requested an interrupt. The highest-priority enabled interrupt generates a number in the UC $Ax$ IV register that can be evaluated or added to the program counter to automatically enter the appropriate software routine. Disabled interrupts do not affect the UC $Ax$ IV value.

Read access of the UC $Ax$ IV register automatically resets the highest-pending Interrupt condition and flag. Write access of the UC $Ax$ IV register clears all pending Interrupt conditions and flags. If another interrupt flag is set, another interrupt is generated immediately after servicing the initial interrupt.

[Example 22-1](#) shows the recommended use of UC $Ax$ IV. The UC $Ax$ IV value is added to the PC to automatically jump to the appropriate routine. The following example is given for eUSCI\_A0.

#### Example 22-1. UC $Ax$ IV Software Example

```
#pragma vector = USCI_A0_VECTOR __interrupt void USCI_A0_ISR(void) {
    switch(__even_in_range(UCA0IV,18)) {
        case 0x00:          // Vector 0: No interrupts
            break;
        case 0x02: ...      // Vector 2: UCRXIFG
            break;
        case 0x04: ...      // Vector 4: UCTXIFG
            break;
        case 0x06: ...      // Vector 6: UCSTTIFG
            break;
        case 0x08: ...      // Vector 8: UCTXCPTIFG
            break;
        default: break;
    }
}
```

### 22.3.16 DMA Operation

In devices with a DMA controller, the eUSCI module can trigger DMA transfers when the transmit buffer UCAXTBUF is empty or when data was received in the UCAXRBUF buffer. The DMA trigger signals correspond to the UCTXIFG transmit interrupt flag and the UCRXIFG receive interrupt flag, respectively. The interrupt functionality must be disabled for the selected DMA triggers with UCTXIE = 0 or UCRXIE = 0.

A DMA read access to UCAXRBUF has the same effects as a CPU (software) read: all error flags (UCRXERR, UCFE, UCPE, UCOE, and UCBRK) are cleared after the read. Thus these errors might go unnoticed.

## 22.4 eUSCI\_A UART Registers

The eUSCI\_A registers applicable in UART mode and their address offsets are listed in [Table 22-7](#). The base address can be found in the device-specific data sheet.

**Table 22-7. eUSCI\_A UART Registers**

Offset	Acronym	Register Name	Section
00h	UCAxCTLW0	eUSCI_Ax Control Word 0	<a href="#">Section 22.4.1</a>
00h	UCAxCTL1	eUSCI_Ax Control 1	
01h	UCAxCTL0	eUSCI_Ax Control 0	
02h	UCAxCTLW1	eUSCI_Ax Control Word 1	<a href="#">Section 22.4.2</a>
06h	UCAxBRW	eUSCI_Ax Baud Rate Control Word	<a href="#">Section 22.4.3</a>
06h	UCAxBR0	eUSCI_Ax Baud Rate Control 0	
07h	UCAxBR1	eUSCI_Ax Baud Rate Control 1	
08h	UCAxMCTLW	eUSCI_Ax Modulation Control Word	<a href="#">Section 22.4.4</a>
0Ah	UCAxSTATW	eUSCI_Ax Status	<a href="#">Section 22.4.5</a>
0Ch	UCAxRXBUF	eUSCI_Ax Receive Buffer	<a href="#">Section 22.4.6</a>
0Eh	UCAxTXBUF	eUSCI_Ax Transmit Buffer	<a href="#">Section 22.4.7</a>
10h	UCAxABCTL	eUSCI_Ax Auto Baud Rate Control	<a href="#">Section 22.4.8</a>
12h	UCAxIRCTL	eUSCI_Ax IrDA Control	<a href="#">Section 22.4.9</a>
12h	UCAxIRTCTL	eUSCI_Ax IrDA Transmit Control	
13h	UCAxIRRCTL	eUSCI_Ax IrDA Receive Control	
1Ah	UCAxIE	eUSCI_Ax Interrupt Enable	<a href="#">Section 22.4.10</a>
1Ch	UCAxIFG	eUSCI_Ax Interrupt Flag	<a href="#">Section 22.4.11</a>
1Eh	UCAxIV	eUSCI_Ax Interrupt Vector	<a href="#">Section 22.4.12</a>

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

## 22.4.1 UCAXCTLW0 Register

eUSCI\_Ax Control Word Register 0

**Figure 22-12. UCAXCTLW0 Register**

15	14	13	12	11	10	9	8
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB	UCMODEx	UCSYNC	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCSELx	UCRXIE	UCBRKIE	UCDORM	UCTXADDR	UCTXBRK	UCSWRST	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
Modify only when UCSWRST = 1							

**Table 22-8. UCAXCTLW0 Register Description**

Bit	Field	Type	Reset	Description
15	UCPEN	RW	0h	Parity enable 0b = Parity disabled 1b = Parity enabled. Parity bit is generated (UCAxTXD) and expected (UCAxRXD). In address-bit multiprocessor mode, the address bit is included in the parity calculation.
14	UCPAR	RW	0h	Parity select. UCPAR is not used when parity is disabled. 0b = Odd parity 1b = Even parity
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCSPB	RW	0h	Stop bit select. Number of stop bits. 0b = One stop bit 1b = Two stop bits
10-9	UCMODEx	RW	0h	eUSCI_A mode. The UCMODEx bits select the asynchronous mode when UCSYNC = 0. 00b = UART mode 01b = Idle-line multiprocessor mode 10b = Address-bit multiprocessor mode 11b = UART mode with automatic baud-rate detection
8	UCSYNC	RW	0h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSELx	RW	0h	eUSCI_A clock source select. These bits select the BRCLK source clock. 00b = UCLK 01b = ACLK 10b = SMCLK 11b = SMCLK
5	UCRXIE	RW	0h	Receive erroneous-character interrupt enable 0b = Erroneous characters rejected and UCRXIFG is not set. 1b = Erroneous characters received set UCRXIFG.
4	UCBRKIE	RW	0h	Receive break character interrupt enable 0b = Received break characters do not set UCRXIFG. 1b = Received break characters set UCRXIFG.

**Table 22-8. UCAXCTLW0 Register Description (continued)**

Bit	Field	Type	Reset	Description
3	UCDORM	RW	0h	Dormant. Puts eUSCI_A into sleep mode. 0b = Not dormant. All received characters set UCRXIFG. 1b = Dormant. Only characters that are preceded by an idle-line or with address bit set UCRXIFG. In UART mode with automatic baud-rate detection, only the combination of a break and synch field sets UCRXIFG.
2	UCTXADDR	RW	0h	Transmit address. Next frame to be transmitted is marked as address, depending on the selected multiprocessor mode. 0b = Next frame transmitted is data. 1b = Next frame transmitted is an address.
1	UCTXBRK	RW	0h	Transmit break. Transmits a break with the next write to the transmit buffer. In UART mode with automatic baud-rate detection, 055h must be written into UCAXTXBUF to generate the required break/synch fields. Otherwise, 0h must be written into the transmit buffer. 0b = Next frame transmitted is not a break. 1b = Next frame transmitted is a break or a break/synch.
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI_A reset released for operation. 1b = Enabled. eUSCI_A logic held in reset state.

#### 22.4.2 UCAXCTLW1 Register

eUSCI\_Ax Control Word Register 1

**Figure 22-13. UCAXCTLW1 Register**

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-1

**Table 22-9. UCAXCTLW1 Register Description**

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1-0	UCGLITx	RW	3h	Deglitch time 00b = Approximately 5 ns 01b = Approximately 20 ns 10b = Approximately 30 ns 11b = Approximately 50 ns

### 22.4.3 UCAXBRW Register

eUSCI\_Ax Baud Rate Control Word Register

**Figure 22-14. UCAXBRW Register**

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
Modify only when UCSWRST = 1							

**Table 22-10. UCAXBRW Register Description**

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Clock prescaler setting of the baud-rate generator

### 22.4.4 UCAXMCTLW Register

eUSCI\_Ax Modulation Control Word Register

**Figure 22-15. UCAXMCTLW Register**

15	14	13	12	11	10	9	8
UCBRSx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCBRFx				Reserved		UCOS16	
rw-0	rw-0	rw-0	rw-0	r0	r0	r0	rw-0
Modify only when UCSWRST = 1							

**Table 22-11. UCAXMCTLW Register Description**

Bit	Field	Type	Reset	Description
15-8	UCBRSx	RW	0h	Second modulation stage select. These bits hold a free modulation pattern for BITCLK.
7-4	UCBRFx	RW	0h	First modulation stage select. These bits determine the modulation pattern for BITCLK16 when UCOS16 = 1. Ignored with UCOS16 = 0. The "Oversampling Baud-Rate Generation" section shows the modulation pattern.
3-1	Reserved	R	0h	Reserved
0	UCOS16	RW	0h	Oversampling mode enabled 0b = Disabled 1b = Enabled

## 22.4.5 UCAXSTATW Register

eUSCI\_Ax Status Register

**Figure 22-16. UCAXSTATW Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE	UCPE	UCBRK	UCRXERR	UCADDR UCIDLE	UCBUSY
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0
Modify only when UCSWRST = 1							

**Table 22-12. UCAXSTATW Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. The UCLISTEN bit selects loopback mode. 0b = Disabled 1b = Enabled. UCAXTXD is internally fed back to the receiver.
6	UCFE	RW	0h	Framing error flag. UCFE is cleared when UCAXRXBUF is read. 0b = No error 1b = Character received with low stop bit
5	UCOE	RW	0h	Overrun error flag. This bit is set when a character is transferred into UCAXRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it does not function correctly. 0b = No error 1b = Overrun error occurred.
4	UCPE	RW	0h	Parity error flag. When UCPEN = 0, UCPE is read as 0. UCPE is cleared when UCAXRXBUF is read. 0b = No error 1b = Character received with parity error
3	UCBRK	RW	0h	Break detect flag. UCBRK is cleared when UCAXRXBUF is read. 0b = No break condition 1b = Break condition occurred.
2	UCRXERR	RW	0h	Receive error flag. This bit indicates a character was received with one or more errors. When UCRXERR = 1, one or more error flags, UCFE, UCPE, or UCOE is also set. UCRXERR is cleared when UCAXRXBUF is read. 0b = No receive errors detected 1b = Receive error detected
1	UCADDR UCIDLE	RW	0h	UCADDR: Address received in address-bit multiprocessor mode. UCADDR is cleared when UCAXRXBUF is read. UCIDLE: Idle line detected in idle-line multiprocessor mode. UCIDLE is cleared when UCAXRXBUF is read. 0b = UCADDR: Received character is data. UCIDLE: No idle line detected 1b = UCADDR: Received character is an address. UCIDLE: Idle line detected
0	UCBUSY	R	0h	eUSCI_A busy. This bit indicates if a transmit or receive operation is in progress. 0b = eUSCI_A inactive 1b = eUSCI_A transmitting or receiving

## 22.4.6 UCAXRXBUF Register

eUSCI\_Ax Receive Buffer Register

**Figure 22-17. UCAXRXBUF Register**

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
UCRXBUFx							
r	r	r	r	r	r	r	r

**Table 22-13. UCAXRXBUF Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCAXRXBUF resets the receive-error bits, the UCADDR or UCIDLE bit, and UCRXIFG. In 7-bit data mode, UCAXRXBUF is LSB justified and the MSB is always reset.

## 22.4.7 UCAXTXBUF Register

eUSCI\_Ax Transmit Buffer Register

**Figure 22-18. UCAXTXBUF Register**

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 22-14. UCAXTXBUF Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted on UCAXTXD. Writing to the transmit data buffer clears UCTXIFG. The MSB of UCAXTXBUF is not used for 7-bit data and is reset.

## 22.4.8 UCAXABCTL Register

eUSCI\_Ax Auto Baud Rate Control Register

**Figure 22-19. UCAXABCTL Register**

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved							
r-0	r-0	rw-0	rw-0	rw-0	rw-0	r-0	rw-0
Modify only when UCSWRST = 1							

**Table 22-15. UCAXABCTL Register Description**

Bit	Field	Type	Reset	Description
15-6	Reserved	R	0h	Reserved
5-4	UCDELIMx	RW	0h	Break/synch delimiter length 00b = 1 bit time 01b = 2 bit times 10b = 3 bit times 11b = 4 bit times
3	UCSTOE	RW	0h	Synch field time out error 0b = No error 1b = Length of synch field exceeded measurable time.
2	UCBTOE	RW	0h	Break time out error 0b = No error 1b = Length of break field exceeded 22 bit times.
1	Reserved	R	0h	Reserved
0	UCABDEN	RW	0h	Automatic baud-rate detect enable 0b = Baud-rate detection disabled. Length of break and synch field is not measured. 1b = Baud-rate detection enabled. Length of break and synch field is measured and baud-rate settings are changed accordingly.

## 22.4.9 UCAXIRCTL Register

eUSCI\_Ax IrDA Control Word Register

**Figure 22-20. UCAXIRCTL Register**

15	14	13	12	11	10	9	8
UCIRRXFLx				UCIRRXPL		UCIRRXFE	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCIRTXPLx				UCIRTXCLK		UCIREN	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
Modify only when UCSWRST = 1							

**Table 22-16. UCAXIRCTL Register Description**

Bit	Field	Type	Reset	Description
15-10	UCIRRXFLx	RW	0h	Receive filter length. The minimum pulse length for receive is given by: $t_{MIN} = (UCIRRXFLx + 4) / (2 \times f_{IRTXCLK})$
9	UCIRRXPL	RW	0h	IrDA receive input UCAXRXD polarity 0b = IrDA transceiver delivers a high pulse when a light pulse is seen. 1b = IrDA transceiver delivers a low pulse when a light pulse is seen.
8	UCIRRXFE	RW	0h	IrDA receive filter enabled 0b = Receive filter disabled 1b = Receive filter enabled
7-2	UCIRTXPLx	RW	0h	Transmit pulse length. Pulse length $t_{PULSE} = (UCIRTXPLx + 1) / (2 \times f_{IRTXCLK})$
1	UCIRTXCLK	RW	0h	IrDA transmit pulse clock select 0b = BRCLK 1b = BITCLK16 when UCOS16 = 1. Otherwise, BRCLK.
0	UCIREN	RW	0h	IrDA encoder and decoder enable 0b = IrDA encoder/decoder disabled 1b = IrDA encoder/decoder enabled

### **22.4.10 UCAXIE Register**

eUSCI\_Ax Interrupt Enable Register

**Figure 22-21. UCAXIE Register**

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved				UCTXCPTIE	UCSTTIE	UCTXIE	UCRXIE
r-0	r-0	r-0	r-0	rw-0	rw-0	rw-0	rw-0

**Table 22-17. UCAXIE Register Description**

Bit	Field	Type	Reset	Description
15-4	Reserved	R	0h	Reserved
3	UCTXCPTIE	RW	0h	Transmit complete interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
2	UCSTTIE	RW	0h	Start bit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

### 22.4.11 UCAXIFG Register

eUSCI\_Ax Interrupt Flag Register

**Figure 22-22. UCAXIFG Register**

15	14	13	12	11	10	9	8
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved				UCTXCPTIFG	UCSTTIFG	UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	rw-0	rw-0	rw-1	rw-0

**Table 22-18. UCAXIFG Register Description**

Bit	Field	Type	Reset	Description
15-4	Reserved	R	0h	Reserved
3	UCTXCPTIFG	RW	0h	Transmit complete interrupt flag. UCTXCPTIFG is set when the entire byte in the internal shift register got shifted out and UCAXTXBUF is empty. 0b = No interrupt pending 1b = Interrupt pending
2	UCSTTIFG	RW	0h	Start bit interrupt flag. UCSTTIFG is set after a Start bit was received 0b = No interrupt pending 1b = Interrupt pending
1	UCTXIFG	RW	1h	Transmit interrupt flag. UCTXIFG is set when UCAXTXBUF empty. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG	RW	0h	Receive interrupt flag. UCRXIFG is set when UCAXRXBUF has received a complete character. 0b = No interrupt pending 1b = Interrupt pending

### **22.4.12 UCAXIV Register**

eUSCI\_Ax Interrupt Vector Register

**Figure 22-23. UCAXIV Register**

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
UCIVx							
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

**Table 22-19. UCAXIV Register Description**

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI_A interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Receive buffer full; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 04h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG 06h = Interrupt Source: Start bit received; Interrupt Flag: UCSTTIFG 08h = Interrupt Source: Transmit complete; Interrupt Flag: UCTXCPTIFG; Interrupt Priority: Lowest

## ***Enhanced Universal Serial Communication Interface (eUSCI) – SPI Mode***

---

---

The enhanced universal serial communication interfaces, eUSCI\_A and eUSCI\_B, support multiple serial communication modes with one hardware module. This chapter discusses the operation of the synchronous peripheral interface (SPI) mode.

Topic	Page
23.1 Enhanced Universal Serial Communication Interfaces (eUSCI_A, eUSCI_B) Overview .....	755
23.2 eUSCI Introduction – SPI Mode .....	755
23.3 eUSCI Operation – SPI Mode .....	757
23.4 eUSCI_A SPI Registers .....	763
23.5 eUSCI_B SPI Registers .....	772

## 23.1 Enhanced Universal Serial Communication Interfaces (eUSCI\_A, eUSCI\_B) Overview

Both the eUSCI\_A and the eUSCI\_B support serial communication in SPI mode.

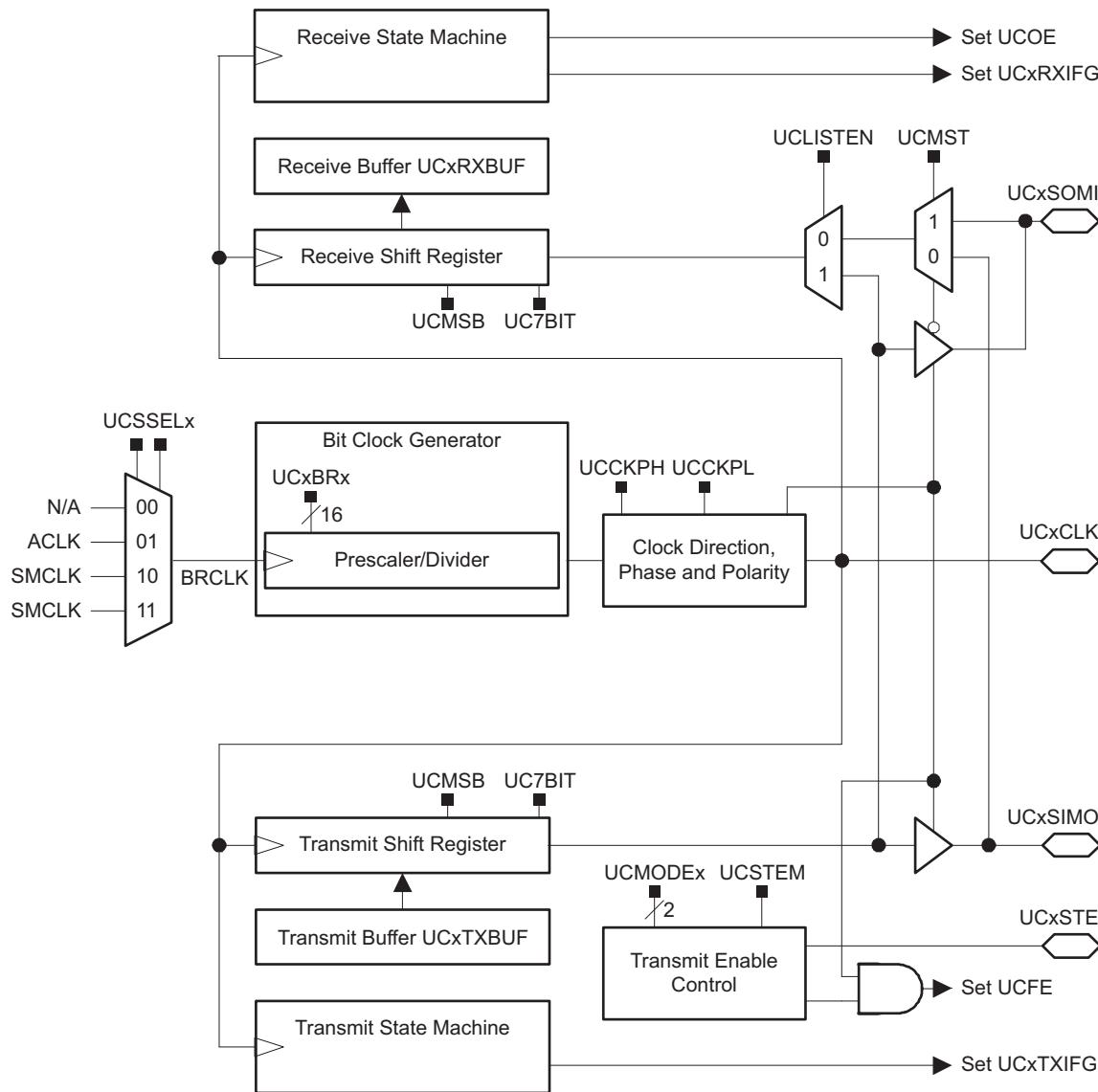
## 23.2 eUSCI Introduction – SPI Mode

In synchronous mode, the eUSCI connects the device to an external system through three or four pins: UCxSIMO, UCxSOMI, UCxCLK, and UCxSTE. SPI mode is selected when the UCSYNC bit is set, and SPI mode (3 pin or 4 pin) is selected with the UCMODEEx bits.

SPI mode features include:

- 7-bit or 8-bit data length
- LSB-first or MSB-first data transmit and receive
- 3-pin and 4-pin SPI operation
- Master or slave modes
- Independent transmit and receive shift registers
- Separate transmit and receive buffer registers
- Continuous transmit and receive operation
- Selectable clock polarity and phase control
- Programmable clock frequency in master mode
- Independent interrupt capability for receive and transmit

[Figure 23-1](#) shows the eUSCI when configured for SPI mode.



**Figure 23-1. eUSCI Block Diagram – SPI Mode**

### 23.3 eUSCI Operation – SPI Mode

In SPI mode, serial data is transmitted and received by multiple devices using a shared clock provided by the master. An additional pin controlled by the master, UCxSTE, is provided to enable a device to receive and transmit data.

Three or four signals are used for SPI data exchange:

- UCxSIMO – slave in, master out  
Master mode: UCxSIMO is the data output line.  
Slave mode: UCxSIMO is the data input line.
- UCxSOMI – slave out, master in  
Master mode: UCxSOMI is the data input line.  
Slave mode: UCxSOMI is the data output line.
- UCxCLK – eUSCI SPI clock  
Master mode: UCxCLK is an output.  
Slave mode: UCxCLK is an input.
- UCxSTE – slave transmit enable

Used in 4-pin mode to allow multiple masters on a single bus. Not used in 3-pin mode. [Table 23-1](#) describes the UCxSTE operation.

**Table 23-1. UCxSTE Operation**

UCMODEx	UCxSTE Active State	UCxSTE	Slave	Master
01	High	0	Inactive	Active
		1	Active	Inactive
10	Low	0	Active	Inactive
		1	Inactive	Active

#### 23.3.1 eUSCI Initialization and Reset

The eUSCI is reset by a Hard Reset or by the UCSWRST bit. After a Hard Reset, the UCSWRST bit is automatically set, keeping the eUSCI in a reset condition. When set, the UCSWRST bit resets the UCRXIE, UCTXIE, UCRXIFG, UCOE, and UCFE bits, and sets the UCTXIFG flag. Clearing UCSWRST releases the eUSCI for operation.

Configuring and reconfiguring the eUSCI module should be done when UCSWRST is set to avoid unpredictable behavior.

---

**NOTE: Initializing or reconfiguring the eUSCI module**

The recommended eUSCI initialization/reconfiguration process is:

1. Set UCSWRST.
  2. Initialize all eUSCI registers with UCSWRST = 1 (including UCxCTL1).
  3. Configure ports.
  4. Clear UCSWRST through software.
  5. Enable interrupts (optional) with UCRXIE or UCTXIE.
-

### 23.3.2 Character Format

The eUSCI module in SPI mode supports 7-bit and 8-bit character lengths selected by the UC7BIT bit. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset. The UCMSB bit controls the direction of the transfer and selects LSB or MSB first.

**NOTE: Default character format**

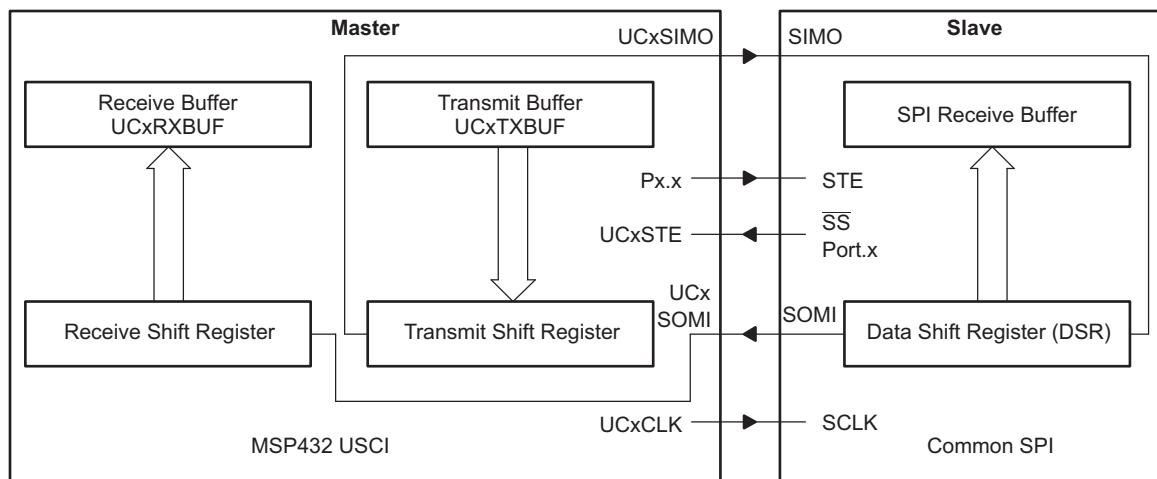
The default SPI character transmission is LSB first. For communication with other SPI interfaces, MSB-first mode may be required.

**NOTE: Character format for figures**

Figures throughout this chapter use MSB-first format.

### 23.3.3 Master Mode

Figure 23-2 shows the eUSCI as a master in both 3-pin and 4-pin configurations. The eUSCI initiates data transfer when data is moved to the transmit data buffer UCxTXBUF. The UCxTXBUF data is moved to the transmit (TX) shift register when the TX shift register is empty, initiating data transfer on UCxSIMO starting with either the MSB or LSB, depending on the UCMSB setting. Data on UCxSOMI is shifted into the receive shift register on the opposite clock edge. When the character is received, the receive data is moved from the receive (RX) shift register to the received data buffer UCxRXBUF and the receive interrupt flag UCRXIFG is set, indicating that the RX or TX operation is complete.



**Figure 23-2. eUSCI Master and External Slave (UCSTEM = 0)**

A set transmit interrupt flag, UCTXIFG, indicates that data has moved from UCxTXBUF to the TX shift register and UCxTXBUF is ready for new data. It does not indicate RX/TX completion.

To receive data into the eUSCI in master mode, data must be written to UCxTXBUF, because receive and transmit operations operate concurrently.

There are two different options for configuring the eUSCI as a 4-pin master, which are described in the next sections:

- The fourth pin is used as input to prevent conflicts with other masters (UCSTEM = 0).
- The fourth pin is used as output to generate a slave enable signal (UCSTEM = 1).

The bit UCSTEM is used to select the corresponding mode.

### 23.3.3.1 4-Pin SPI Master Mode (UCSTEM = 0)

In 4-pin master mode with UCSTEM = 0, UCxSTE is a digital input that can be used to prevent conflicts with another master and controls the master as described in [Table 23-1](#). When UCxSTE is in the master-inactive state and UCSTEM = 0:

- UCxSIMO and UCxCLK are set to inputs and no longer drive the bus.
- The error bit UCFE is set, indicating a communication integrity violation to be handled by the user.
- The internal state machines are reset and the shift operation is aborted.

If data is written into UCxTXBUF while the master is held inactive by UCxSTE, it is transmit as soon as UCxSTE transitions to the master-active state. If an active transfer is aborted by UCxSTE transitioning to the master-inactive state, the data must be rewritten into UCxTXBUF to be transferred when UCxSTE transitions back to the master-active state. The UCxSTE input signal is not used in 3-pin master mode.

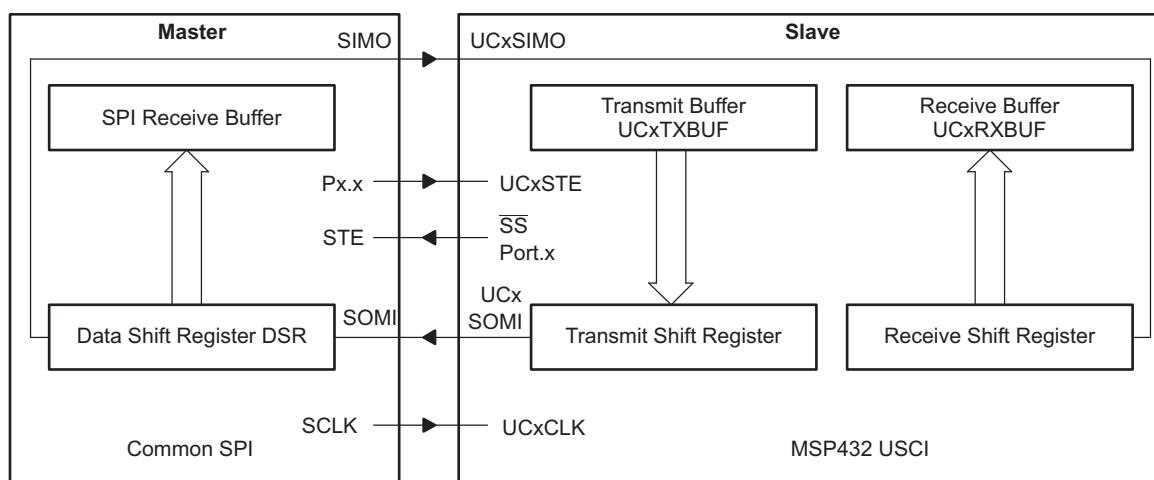
### 23.3.3.2 4-Pin SPI Master Mode (UCSTEM = 1)

If UCSTEM = 1 in 4-pin master mode, UCxSTE is a digital output. In this mode the slave enable signal for a single slave is automatically generated on UCxSTE. The corresponding behavior can be seen in [Figure 23-4](#).

If multiple slaves are desired, this feature is not applicable and the software needs to use general purpose I/O pins instead to generate STE signals for each slave individually.

### 23.3.4 Slave Mode

[Figure 23-3](#) shows the eUSCI as a slave in both 3-pin and 4-pin configurations. UCxCLK is used as the input for the SPI clock and must be supplied by the external master. The data-transfer rate is determined by this clock and not by the internal bit clock generator. Data written to UCxTXBUF and moved to the TX shift register before the start of UCxCLK is transmitted on UCxSOMI. Data on UCxSIMO is shifted into the receive shift register on the opposite edge of UCxCLK and moved to UCxRXBUF when the set number of bits are received. When data is moved from the RX shift register to UCxRXBUF, the UCRXIFG interrupt flag is set, indicating that data has been received. The overrun error bit UCOE is set when the previously received data is not read from UCxRXBUF before new data is moved to UCxRXBUF.



**Figure 23-3. eUSCI Slave and External Master**

### 23.3.4.1 4-Pin SPI Slave Mode

In 4-pin slave mode, UCxSTE is a digital input used by the slave to enable the transmit and receive operations and is driven by the SPI master. When UCxSTE is in the slave-active state, the slave operates normally. When UCxSTE is in the slave-inactive state:

- Any receive operation in progress on UCxSIMO is halted.
- UCxSOMI is set to the input direction.
- The shift operation is halted until the UCxSTE line transitions into the slave transmit active state.

The UCxSTE input signal is not used in 3-pin slave mode.

### 23.3.5 SPI Enable

When the eUSCI module is enabled by clearing the UCSWRST bit, it is ready to receive and transmit. In master mode, the bit clock generator is ready, but is not clocked nor producing any clocks. In slave mode, the bit clock generator is disabled and the clock is provided by the master.

A transmit or receive operation is indicated by UCBUSY = 1.

A Hard Reset or set UCSWRST bit disables the eUSCI immediately and any active transfer is terminated.

#### 23.3.5.1 Transmit Enable

In master mode, writing to UCxTXBUF activates the bit clock generator, and the data begins to transmit.

In slave mode, transmission begins when a master provides a clock and, in 4-pin mode, when the UCxSTE is in the slave-active state.

#### 23.3.5.2 Receive Enable

The SPI receives data when a transmission is active. Receive and transmit operations operate concurrently.

### 23.3.6 Serial Clock Control

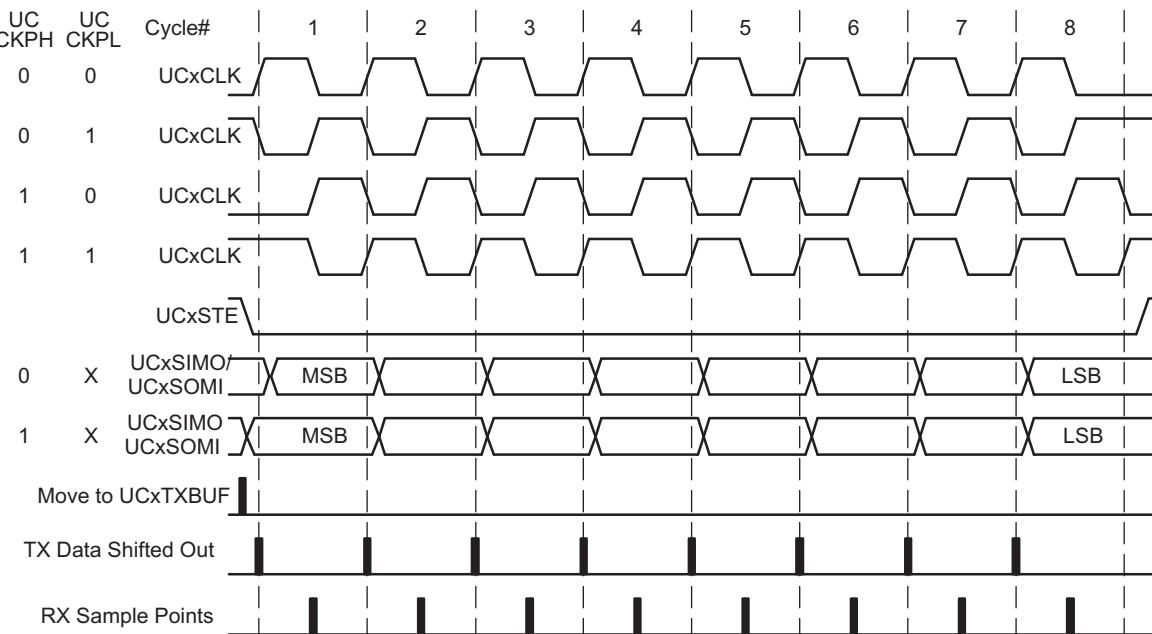
UCxCLK is provided by the master on the SPI bus. When UCMST = 1, the bit clock is provided by the eUSCI bit clock generator on the UCxCLK pin. The clock used to generate the bit clock is selected with the UCSSELx bits. When UCMST = 0, the eUSCI clock is provided on the UCxCLK pin by the master, the bit clock generator is not used, and the UCSSELx bits are don't care. The SPI receiver and transmitter operate in parallel and use the same clock source for data transfer.

The 16-bit value of UCBRx in the bit rate control registers (UCxxBR1 and UCxxBR0) is the division factor of the eUSCI clock source, BRCLK. The maximum bit clock that can be generated in master mode is BRCLK. Modulation is not used in SPI mode, and UCAxMCTLW should be cleared when using SPI mode for eUSCI\_A. The UCAxCLK or UCBxCLK frequency is given by:

$$f_{\text{BitClock}} = f_{\text{BRCLK}} / UCBRx$$

#### 23.3.6.1 Serial Clock Polarity and Phase

The polarity and phase of UCxCLK are independently configured with the UCCKPL and UCCKPH control bits of the eUSCI. [Figure 23-4](#) shows the timing for each case.



**Figure 23-4. eUSCI SPI Timing With UCMSB = 1**

### 23.3.7 Using the SPI Mode With Low-Power Modes

The eUSCI module is not functional when the device is in the LPM3, LPM4, or LPMx.5 modes of operation. However, the application can make use of the FORCE\_LPM\_ENTRY bit in the PCMCTL1 register to determine whether the entry to low-power mode should be aborted if the eUSCI is active, or if the device can continue low power entry regardless. The latter option is useful if the eUSCI is transmitting or receiving data at a very slow rate and the application should enter low-power mode at the expense of a packet of data being lost. Refer to the *Power Control Manager (PCM)* chapter for more details.

### 23.3.8 SPI Interrupts

#### 23.3.8.1 SPI Transmit Interrupt Operation

The UCTXIFG interrupt flag is set by the transmitter to indicate that UCxTXBUF is ready to accept another character. An interrupt request is generated if UCTXIE is set. UCTXIFG is automatically reset if a character is written to UCxTXBUF. UCTXIFG is set after a Hard Reset or when UCSWRST = 1. UCTXIE is reset after a Hard Reset or when UCSWRST = 1.

---

**NOTE: Writing to UCxTXBUF in SPI mode**

Data written to UCxTXBUF when UCTXIFG = 0 may result in erroneous data transmission.

---

#### 23.3.8.2 SPI Receive Interrupt Operation

The UCRXIFG interrupt flag is set each time a character is received and loaded into UCxRXBUF. An interrupt request is generated if UCRXIE is set. UCRXIFG and UCRXIE are reset by a Hard Reset or when UCSWRST = 1. UCRXIFG is automatically reset when UCxRXBUF is read.

### 23.3.8.3 UCxIV, Interrupt Vector Generator

The eUSCI interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register UCxIV is used to determine which flag requested an interrupt. The highest-priority enabled interrupt generates a number in the UCxIV register that can be evaluated by the interrupt service routine to automatically enter the appropriate software routine. Disabled interrupts do not affect the UCxIV value.

Any access, read or write, of the UCxIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt.

## 23.4 eUSCI\_A SPI Registers

The eUSCI\_A registers applicable in SPI mode and their address offsets are listed in [Table 23-2](#). The base addresses can be found in the device-specific data sheet.

**Table 23-2. eUSCI\_A SPI Registers**

Offset	Acronym	Register Name	Section
00h	UCAxCTLW0	eUSCI_Ax Control Word 0	<a href="#">Section 23.4.1</a>
00h	UCAxCTL1	eUSCI_Ax Control 1	
01h	UCAxCTL0	eUSCI_Ax Control 0	
06h	UCAxBRW	eUSCI_Ax Bit Rate Control Word	<a href="#">Section 23.4.2</a>
06h	UCAxBR0	eUSCI_Ax Bit Rate Control 0	
07h	UCAxBR1	eUSCI_Ax Bit Rate Control 1	
0Ah	UCAxSTATW	eUSCI_Ax Status	<a href="#">Section 23.4.3</a>
0Ch	UCAxRXBUF	eUSCI_Ax Receive Buffer	<a href="#">Section 23.4.4</a>
0Eh	UCAxTXBUF	eUSCI_Ax Transmit Buffer	<a href="#">Section 23.4.5</a>
1Ah	UCAxIE	eUSCI_Ax Interrupt Enable	<a href="#">Section 23.4.6</a>
1Ch	UCAxIFG	eUSCI_Ax Interrupt Flag	<a href="#">Section 23.4.7</a>
1Eh	UCAxIV	eUSCI_Ax Interrupt Vector	<a href="#">Section 23.4.8</a>

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 23.4.1 UCAXCTLW0 Register

eUSCI\_Ax Control Register 0

**Figure 23-5. UCAXCTLW0 Register**

15	14	13	12	11	10	9	8
UCCKPH	UCCKPL	UCMSB	UC7BIT	UCMST	UCMODEx	UCSYNC	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
UCSELx		Reserved			UCSTEM	UCSWRST	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
Modify only when UCSWRST = 1.							

**Table 23-3. UCAXCTLW0 Register Description**

Bit	Field	Type	Reset	Description
15	UCCKPH	RW	0h	Clock phase select 0b = Data is changed on the first UCLK edge and captured on the following edge. 1b = Data is captured on the first UCLK edge and changed on the following edge.
14	UCCKPL	RW	0h	Clock polarity select 0b = The inactive state is low. 1b = The inactive state is high.
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCMST	RW	0h	Master mode select 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. 00b = 3-pin SPI 01b = 4-pin SPI with UCxSTE active high: Slave enabled when UCxSTE = 1 10b = 4-pin SPI with UCxSTE active low: Slave enabled when UCxSTE = 0 11b = I2C mode
8	UCSYNC	RW	0h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSELx	RW	0h	eUSCI clock source select. These bits select the BRCLK source clock in master mode. UCxCLK is always used in slave mode. 00b = Reserved 01b = ACLK 10b = SMCLK 11b = SMCLK
5-2	Reserved	R	0h	Reserved
1	UCSTEM	RW	0h	STE mode select in master mode. This byte is ignored in slave or 3-wire mode. 0b = STE pin is used to prevent conflicts with other masters 1b = STE pin is used to generate the enable signal for a 4-wire slave
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI reset released for operation. 1b = Enabled. eUSCI logic held in reset state.

### **23.4.2 UCAXBRW Register**

eUSCI\_Ax Bit Rate Control Register 1

**Figure 23-6. UCAXBRW Register**

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
Modify only when UCSWRST = 1.							

**Table 23-4. UCAXBRW Register Description**

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler setting

### 23.4.3 UCAXSTATW Register

eUSCI\_Ax Status Register

**Figure 23-7. UCAXSTATW Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE		Reserved			UCBUSY
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0
Modify only when UCSWRST = 1.							

**Table 23-5. UCAXSTATW Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. The UCLISTEN bit selects loopback mode. 0b = Disabled 1b = Enabled. The transmitter output is internally fed back to the receiver.
6	UCFE	RW	0h	Framing error flag. This bit indicates a bus conflict in 4-wire master mode. UCFE is not used in 3-wire master or any slave mode. 0b = No error 1b = Bus conflict occurred
5	UCOE	RW	0h	Overrun error flag. This bit is set when a character is transferred into UCxRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it does not function correctly. 0b = No error 1b = Overrun error occurred
4-1	Reserved	RW	0h	Reserved
0	UCBUSY	R	0h	eUSCI busy. This bit indicates if a transmit or receive operation is in progress. 0b = eUSCI inactive 1b = eUSCI transmitting or receiving

### **23.4.4 UCAXRXBUF Register**

eUSCI\_Ax Receive Buffer Register

**Figure 23-8. UCAXRXBUF Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCRXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 23-6. UCAXRXBUF Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCxRXBUF resets the receive-error bits and UCRXIFG. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset.

### 23.4.5 UCAXTBUF Register

eUSCI\_Ax Transmit Buffer Register

**Figure 23-9. UCAXTBUF Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 23-7. UCAXTBUF Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears UCTXIFG. The MSB of UCxTBUF is not used for 7-bit data and is reset.

### **23.4.6 UCAXIE Register**

eUSCI\_Ax Interrupt Enable Register

**Figure 23-10. UCAXIE Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

**Table 23-8. UCAXIE Register Description**

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

### 23.4.7 UCAXIFG Register

eUSCI\_Ax Interrupt Flag Register

**Figure 23-11. UCAXIFG Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-0

**Table 23-9. UCAXIFG Register Description**

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIFG	RW	1h	Transmit interrupt flag. UCTXIFG is set when UCxxTXBUF empty. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG	RW	0h	Receive interrupt flag. UCRXIFG is set when UCxxRXBUF has received a complete character. 0b = No interrupt pending 1b = Interrupt pending

### **23.4.8 UCAXIV Register**

eUSCI\_Ax Interrupt Vector Register

**Figure 23-12. UCAXIV Register**

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
UCIVx							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

**Table 23-10. UCAXIV Register Description**

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI interrupt vector value 000h = No interrupt pending 002h = Interrupt Source: Data received; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 004h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG; Interrupt Priority: Lowest

## 23.5 eUSCI\_B SPI Registers

The eUSCI\_B registers applicable in SPI mode and their address offsets are listed in [Table 23-11](#). The base addresses can be found in the device-specific data sheet.

**Table 23-11. eUSCI\_B SPI Registers**

Offset	Acronym	Register Name	Section
00h	UCBxCTLW0	eUSCI_Bx Control Word 0	<a href="#">Section 23.5.1</a>
00h	UCBxCTL1	eUSCI_Bx Control 1	
01h	UCBxCTL0	eUSCI_Bx Control 0	
06h	UCBxBRW	eUSCI_Bx Bit Rate Control Word	<a href="#">Section 23.5.2</a>
06h	UCBxBR0	eUSCI_Bx Bit Rate Control 0	
07h	UCBxBR1	eUSCI_Bx Bit Rate Control 1	
08h	UCBxSTATW	eUSCI_Bx Status	<a href="#">Section 23.5.3</a>
0Ch	UCBxRXBUF	eUSCI_Bx Receive Buffer	<a href="#">Section 23.5.4</a>
0Eh	UCBxTXBUF	eUSCI_Bx Transmit Buffer	<a href="#">Section 23.5.5</a>
2Ah	UCBxIE	eUSCI_Bx Interrupt Enable	<a href="#">Section 23.5.6</a>
2Ch	UCBxIFG	eUSCI_Bx Interrupt Flag	<a href="#">Section 23.5.7</a>
2Eh	UCBxIV	eUSCI_Bx Interrupt Vector	<a href="#">Section 23.5.8</a>

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

### 23.5.1 UCBxCTLW0 Register

eUSCI\_Bx Control Register 0

**Figure 23-13. UCBxCTLW0 Register**

15	14	13	12	11	10	9	8
UCCKPH	UCCKPL	UCMSB	UC7BIT	UCMST	UCMODEx	UCSYNC	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
7	6	5	4	3	2	1	0
UCSSELx		Reserved			UCSTEM	UCSWRST	
rw-1	rw-1	r0	rw-0	rw-0	rw-0	rw-0	rw-1
Modify only when UCSWRST = 1.							

**Table 23-12. UCBxCTLW0 Register Description**

Bit	Field	Type	Reset	Description
15	UCCKPH	RW	0h	Clock phase select 0b = Data is changed on the first UCLK edge and captured on the following edge. 1b = Data is captured on the first UCLK edge and changed on the following edge.
14	UCCKPL	RW	0h	Clock polarity select 0b = The inactive state is low. 1b = The inactive state is high.
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCMST	RW	0h	Master mode select 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. 00b = 3-pin SPI 01b = 4-pin SPI with UCxSTE active high: Slave enabled when UCxSTE = 1 10b = 4-pin SPI with UCxSTE active low: Slave enabled when UCxSTE = 0 11b = I2C mode
8	UCSYNC	RW	1h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSSELx	RW	3h	eUSCI clock source select. These bits select the BRCLK source clock in master mode. UCxCLK is always used in slave mode. 00b = Reserved 01b = ACLK 10b = SMCLK 11b = SMCLK
5-2	Reserved	R	0h	Reserved
1	UCSTEM	RW	0h	STE mode select in master mode. This byte is ignored in slave or 3-wire mode. 0b = STE pin is used to prevent conflicts with other masters 1b = STE pin is used to generate the enable signal for a 4-wire slave
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI reset released for operation. 1b = Enabled. eUSCI logic held in reset state.

### 23.5.2 UCBxBRW Register

eUSCI\_Bx Bit Rate Control Register 1

**Figure 23-14. UCBxBRW Register**

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
Modify only when UCSWRST = 1.							

**Table 23-13. UCBxBRW Register Description**

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler setting.

### 23.5.3 UCBxSTATW Register

eUSCI\_Bx Status Register

**Figure 23-15. UCBxSTATW Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE		Reserved			UCBUSY
rw-0	rw-0	rw-0	r0	r0	r0	r0	r-0
Modify only when UCSWRST = 1.							

**Table 23-14. UCBxSTATW Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7	UCLISTEN	RW	0h	Listen enable. The UCLISTEN bit selects loopback mode. 0b = Disabled 1b = Enabled. The transmitter output is internally fed back to the receiver.
6	UCFE	RW	0h	Framing error flag. This bit indicates a bus conflict in 4-wire master mode. UCFE is not used in 3-wire master or any slave mode. 0b = No error 1b = Bus conflict occurred
5	UCOE	RW	0h	Overrun error flag. This bit is set when a character is transferred into UCxRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it does not function correctly. 0b = No error 1b = Overrun error occurred
4-1	Reserved	R	0h	Reserved
0	UCBUSY	R	0h	eUSCI busy. This bit indicates if a transmit or receive operation is in progress. 0b = eUSCI inactive 1b = eUSCI transmitting or receiving

### **23.5.4 UCBxRXBUF Register**

eUSCI\_Bx Receive Buffer Register

**Figure 23-16. UCBxRXBUF Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
UCRXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 23-15. UCBxRXBUF Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCxRXBUF resets the receive-error bits and UCRXIFG. In 7-bit data mode, UCxRXBUF is LSB justified and the MSB is always reset.

### **23.5.5 UCBxTXBUF Register**

eUSCI\_Bx Transmit Buffer Register

**Figure 23-17. UCBxTXBUF Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 23-16. UCBxTXBUF Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears UCTXIFG. The MSB of UCxTXBUF is not used for 7-bit data and is reset.

### 23.5.6 UCBxIE Register

eUSCI\_Bx Interrupt Enable Register

**Figure 23-18. UCBxIE Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

**Table 23-17. UCBxIE Register Description**

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

### 23.5.7 UCBxIFG Register

eUSCI\_Bx Interrupt Flag Register

**Figure 23-19. UCBxIFG Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved							
r-0	r-0	r-0	r-0	r-0	r-0	rw-1	rw-0

**Table 23-18. UCBxIFG Register Description**

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIFG	RW	1h	Transmit interrupt flag. UCTXIFG is set when UCxxTXBUF empty. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG	RW	0h	Receive interrupt flag. UCRXIFG is set when UCxxRXBUF has received a complete character. 0b = No interrupt pending 1b = Interrupt pending

### 23.5.8 UCBxIV Register

eUSCI\_Bx Interrupt Vector Register

**Figure 23-20. UCBxIV Register**

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
UCIVx							
r0	r0	r0	r-0	r-0	r-0	r-0	r0

**Table 23-19. UCBxIV Register Description**

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI interrupt vector value 0000h = No interrupt pending 0002h = Interrupt Source: Data received; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 0004h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG; Interrupt Priority: Lowest

## ***Enhanced Universal Serial Communication Interface (eUSCI) – I<sup>2</sup>C Mode***

---

---

The enhanced universal serial communication interface B (eUSCI\_B) supports multiple serial communication modes with one hardware module. This chapter discusses the operation of the I<sup>2</sup>C mode.

Topic	Page
24.1 Enhanced Universal Serial Communication Interface B (eUSCI_B) Overview .....	779
24.2 eUSCI_B Introduction – I <sup>2</sup> C Mode .....	779
24.3 eUSCI_B Operation – I <sup>2</sup> C Mode .....	780
24.4 eUSCI_B I2C Registers .....	799

## 24.1 Enhanced Universal Serial Communication Interface B (eUSCI\_B) Overview

The eUSCI\_B module supports two serial communication modes:

- I<sup>2</sup>C mode
- SPI mode

If more than one eUSCI\_B module is implemented on one device, those modules are named with incrementing numbers. For example, if one device has two eUSCI\_B modules, they are named eUSCI0\_B and eUSCI1\_B.

## 24.2 eUSCI\_B Introduction – I<sup>2</sup>C Mode

In I<sup>2</sup>C mode, the eUSCI\_B module provides an interface between the device and I<sup>2</sup>C-compatible devices connected by the two-wire I<sup>2</sup>C serial bus. External components attached to the I<sup>2</sup>C bus serially transmit or receive serial data to or from the eUSCI\_B module through the 2-wire I<sup>2</sup>C interface.

The eUSCI\_B I<sup>2</sup>C mode features include:

- 7-bit and 10-bit device addressing modes
- General call
- START, RESTART, and STOP
- Multi-master transmitter/receiver mode
- Slave receiver/transmitter mode
- Support for standard mode up to 100 kbps, fast mode up to 400 kbps, and fast mode plus up to 1 Mbps
- Programmable UCxCLK frequency in master mode
- Designed for low power
- 8-bit byte counter with interrupt capability and automatic STOP assertion
- Up to four hardware slave addresses, each having its own interrupt and DMA trigger
- Mask register for slave address and address received interrupt
- Clock low time-out interrupt to avoid bus stalls

[Figure 24-1](#) shows the eUSCI\_B when configured in I<sup>2</sup>C mode.

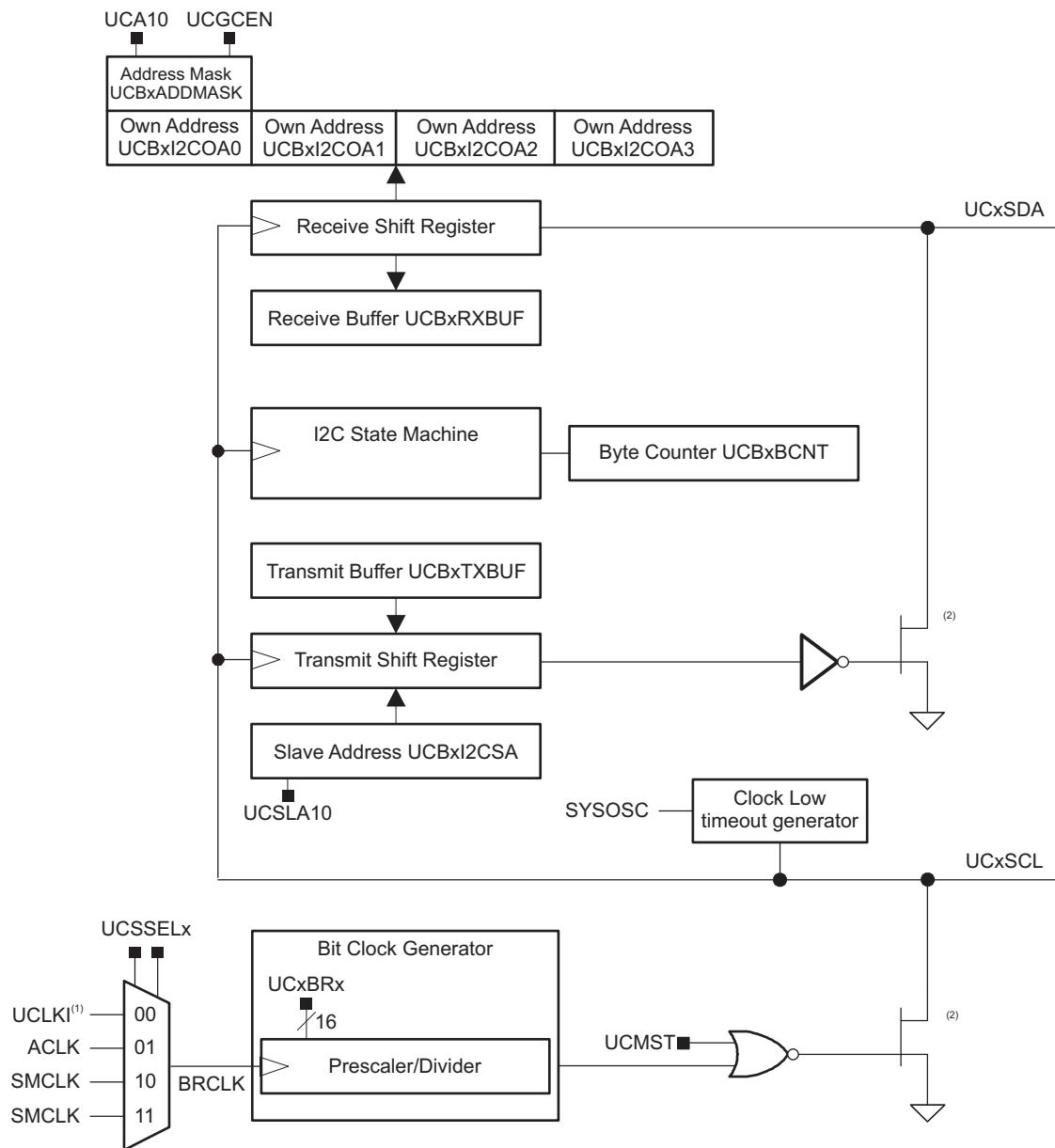
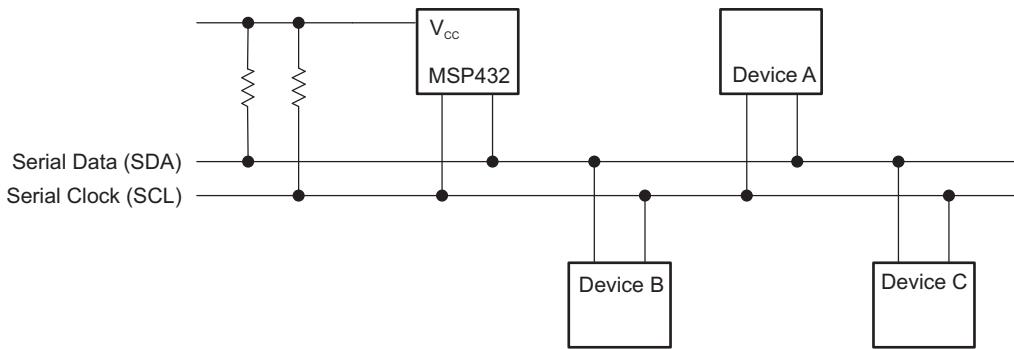


Figure 24-1. eUSCI\_B Block Diagram – I<sup>2</sup>C Mode

### 24.3 eUSCI\_B Operation – I<sup>2</sup>C Mode

The I<sup>2</sup>C mode supports any slave or master I<sup>2</sup>C-compatible device. Figure 24-2 shows an example of an I<sup>2</sup>C bus. Each I<sup>2</sup>C device is recognized by a unique address and can operate as either a transmitter or a receiver. A device connected to the I<sup>2</sup>C bus can be considered as the master or the slave when performing data transfers. A master initiates a data transfer and generates the clock signal SCL. Any device addressed by a master is considered a slave.

I<sup>2</sup>C data is communicated using the serial data (SDA) pin and the serial clock (SCL) pin. Both SDA and SCL are bidirectional and must be connected to a positive supply voltage using a pullup resistor.



**Figure 24-2. I<sup>2</sup>C Bus Connection Diagram**

---

**NOTE: SDA and SCL levels**

The SDA and SCL pins must not be pulled up above the device V<sub>cc</sub> level.

---

#### 24.3.1 eUSCI\_B Initialization and Reset

The eUSCI\_B is reset by a Hard Reset or by setting the UCSWRST bit. After a Hard Reset, the UCSWRST bit is automatically set, keeping the eUSCI\_B in a reset condition. To select I<sup>2</sup>C operation, the UCMODEEx bits must be set to 11. After module initialization, it is ready for transmit or receive operation. Clearing UCSWRST releases the eUSCI\_B for operation.

Configuring and reconfiguring the eUSCI\_B module should be done when UCSWRST is set to avoid unpredictable behavior. Setting UCSWRST in I<sup>2</sup>C mode has the following effects:

- I<sup>2</sup>C communication stops.
- SDA and SCL are high impedance.
- UCBxSTAT, bits 15-9 and 6-4 are cleared.
- Registers UCBxIE and UCBxIFG are cleared.
- All other bits and registers remain unchanged.

---

**NOTE: Initializing or reconfiguring the eUSCI\_B module**

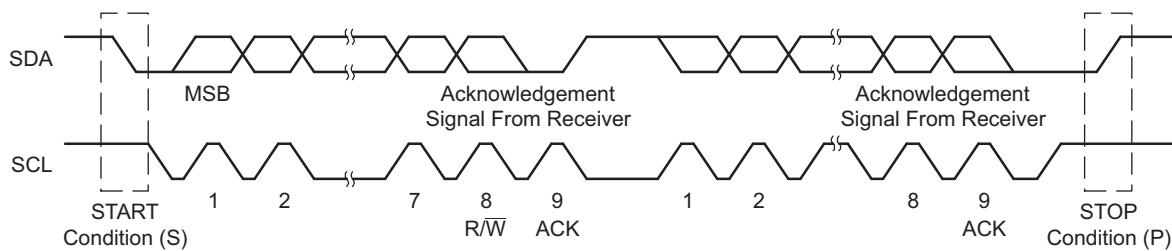
The recommended eUSCI\_B initialization and reconfiguration process is:

1. Set UCSWRST.
  2. Initialize all eUSCI\_B registers with UCSWRST = 1 (including UCxCTL1).
  3. Configure ports.
  4. Clear UCSWRST in software.
  5. Enable interrupts (optional).
- 

#### 24.3.2 I<sup>2</sup>C Serial Data

One clock pulse is generated by the master device for each data bit transferred. The I<sup>2</sup>C mode operates with byte data. Data is transferred MSB first (see [Figure 24-3](#)).

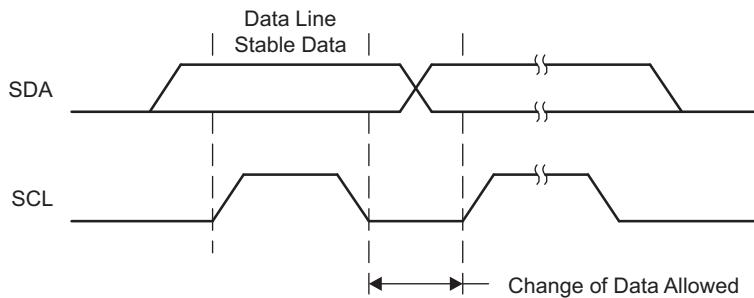
The first byte after a START condition consists of a 7-bit slave address and the R/W bit. When R/W = 0, the master transmits data to a slave. When R/W = 1, the master receives data from a slave. The ACK bit is sent from the receiver after each byte on the ninth SCL clock.



**Figure 24-3. I<sup>2</sup>C Module Data Transfer**

START and STOP conditions are generated by the master (see [Figure 24-3](#)). A START condition is a high-to-low transition on the SDA line while SCL is high. A STOP condition is a low-to-high transition on the SDA line while SCL is high. The bus busy bit, UCBBUSY, is set after a START and cleared after a STOP.

Data on SDA must be stable during the high period of SCL (see [Figure 24-4](#)). The high and low state of SDA can change only when SCL is low, otherwise START or STOP conditions are generated.



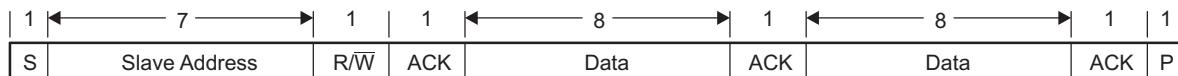
**Figure 24-4. Bit Transfer on I<sup>2</sup>C Bus**

### 24.3.3 I<sup>2</sup>C Addressing Modes

The I<sup>2</sup>C mode supports 7-bit and 10-bit addressing modes.

#### 24.3.3.1 7-Bit Addressing

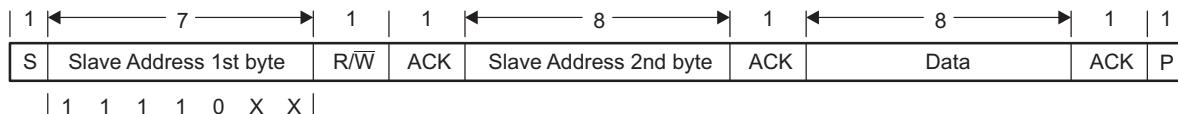
In the 7-bit addressing format (see [Figure 24-5](#)), the first byte is the 7-bit slave address and the R/W bit. The ACK bit is sent from the receiver after each byte.



**Figure 24-5. I<sup>2</sup>C Module 7-Bit Addressing Format**

#### 24.3.3.2 10-Bit Addressing

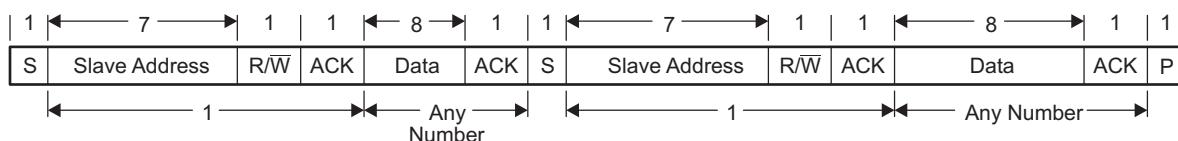
In the 10-bit addressing format (see [Figure 24-6](#)), the first byte is made up of 11110b plus the two MSBs of the 10-bit slave address and the R/W bit. The ACK bit is sent from the receiver after each byte. The next byte is the remaining eight bits of the 10-bit slave address, followed by the ACK bit and the 8-bit data. See [I<sup>2</sup>C Slave 10-bit Addressing Mode](#) and [I<sup>2</sup>C Master 10-bit Addressing Mode](#) for details how to use the 10-bit addressing mode with the eUSCI\_B module.



**Figure 24-6. I<sup>2</sup>C Module 10-Bit Addressing Format**

#### 24.3.3.3 Repeated Start Conditions

The direction of data flow on SDA can be changed by the master, without first stopping a transfer, by issuing a repeated START condition. This is called a RESTART. After a RESTART is issued, the slave address is again sent out with the new data direction specified by the R/W bit. [Figure 24-7](#) shows the RESTART condition.



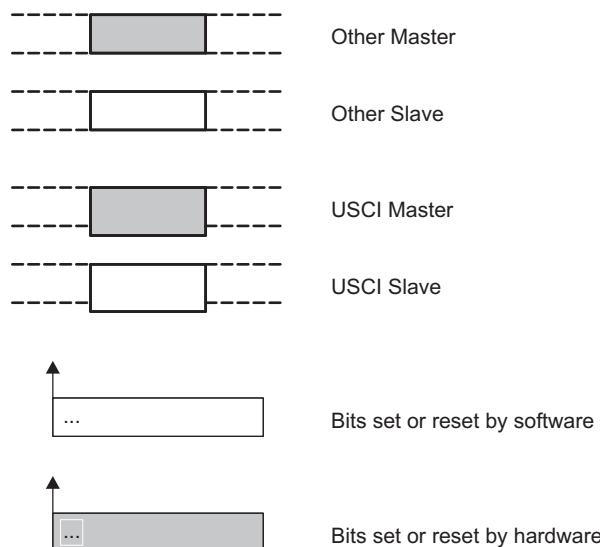
**Figure 24-7. I<sup>2</sup>C Module Addressing Format With Repeated START Condition**

#### 24.3.4 I<sup>2</sup>C Module Operating Modes

In I<sup>2</sup>C mode, the eUSCI\_B module can operate in master transmitter, master receiver, slave transmitter, or slave receiver mode. The modes are discussed in the following sections. Time lines are used to illustrate the modes.

[Figure 24-8](#) shows how to interpret the time-line figures. Data transmitted by the master is represented by grey rectangles; data transmitted by the slave is represented by white rectangles. Data transmitted by the eUSCI\_B module, either as master or slave, is shown by rectangles that are taller than the others.

Actions taken by the eUSCI\_B module are shown in grey rectangles with an arrow indicating where in the data stream the action occurs. Actions that must be handled with software are indicated with white rectangles with an arrow pointing to where in the data stream the action must take place.



**Figure 24-8. I<sup>2</sup>C Time-Line Legend**

#### 24.3.4.1 Slave Mode

The eUSCI\_B module is configured as an I<sup>2</sup>C slave by selecting the I<sup>2</sup>C mode with UCMODEx = 11 and UCSYNC = 1 and clearing the UCMST bit.

Initially, the eUSCI\_B module must be configured in receiver mode by clearing the UCTR bit to receive the I<sup>2</sup>C address. Afterwards, transmit and receive operations are controlled automatically, depending on the R/W bit received together with the slave address.

The eUSCI\_B slave address is programmed with the UCBxI2COA0 register. Support for multiple slave addresses is explained in [Section 24.3.8](#). When UCA10 = 0, 7-bit addressing is selected. When UCA10 = 1, 10-bit addressing is selected. The UCGCEN bit selects if the slave responds to a general call.

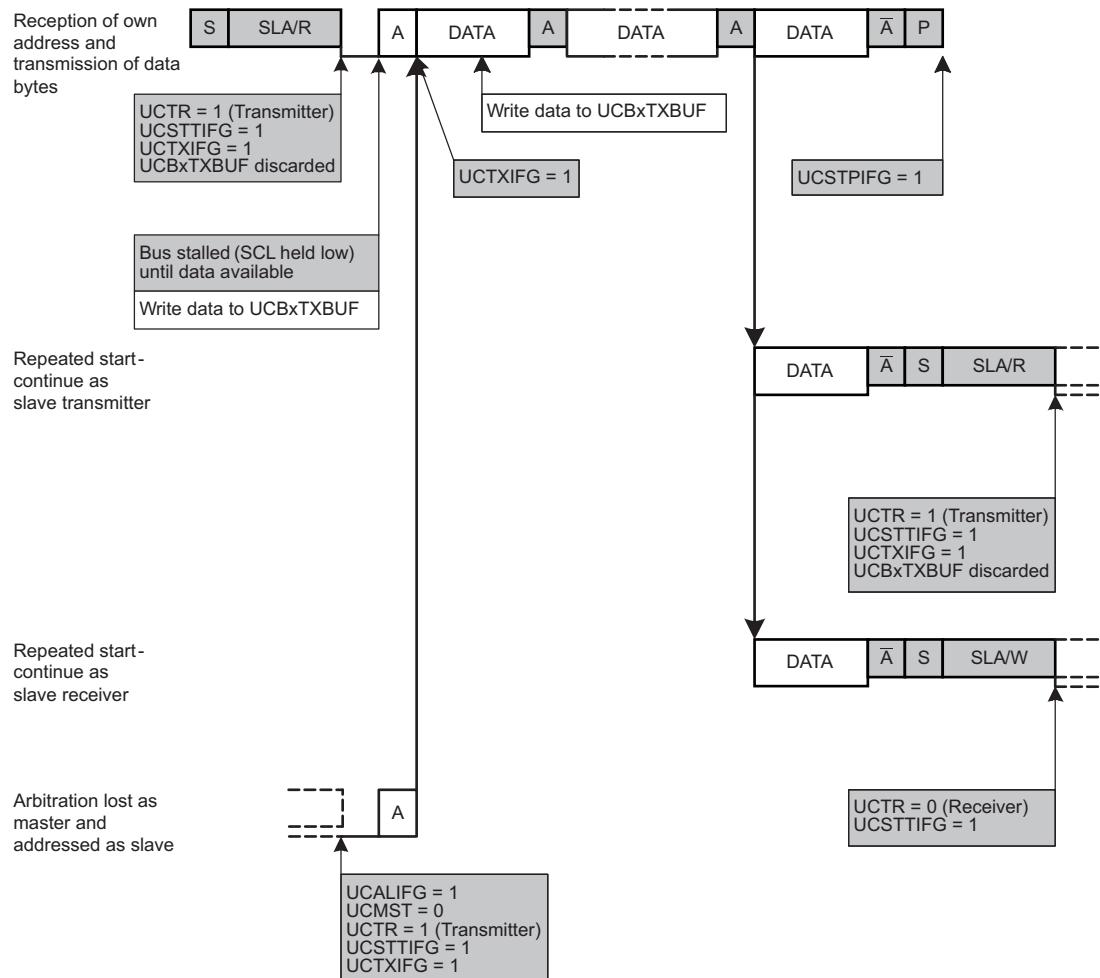
When a START condition is detected on the bus, the eUSCI\_B module receives the transmitted address and compares it against its own address stored in UCBxI2COA0. The UCSTTIFG flag is set when address received matches the eUSCI\_B slave address.

##### 24.3.4.1.1 I<sup>2</sup>C Slave Transmitter Mode

Slave transmitter mode is entered when the slave address transmitted by the master is identical to its own address with a set R/W bit. The slave transmitter shifts the serial data out on SDA with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it does hold SCL low while intervention of the CPU is required after a byte has been transmitted.

If the master requests data from the slave, the eUSCI\_B module is automatically configured as a transmitter and UCTR and UCTXIFG0 become set. The SCL line is held low until the first data to be sent is written into the transmit buffer UCBxTXBUF. Then the address is acknowledged and the data is transmitted. As soon as the data is transferred into the shift register, the UCTXIFG0 is set again. After the data is acknowledged by the master, the next data byte written into UCBxTXBUF is transmitted or, if the buffer is empty, the bus is stalled during the acknowledge cycle by holding SCL low until new data is written into UCBxTXBUF. If the master sends a NACK followed by a STOP condition, the UCSTPIFG flag is set. If the NACK is followed by a repeated START condition, the eUSCI\_B I<sup>2</sup>C state machine returns to its address-reception state.

[Figure 24-9](#) shows the slave transmitter operation.



**Figure 24-9. I<sup>2</sup>C Slave Transmitter Mode**

#### 24.3.4.1.2 I<sup>2</sup>C Slave Receiver Mode

Slave receiver mode is entered when the slave address transmitted by the master is identical to its own address and a cleared R/W bit is received. In slave receiver mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it can hold SCL low if intervention of the CPU is required after a byte has been received.

If the slave receives data from the master, the eUSCI\_B module is automatically configured as a receiver and UCTR is cleared. After the first data byte is received, the receive interrupt flag UCRXIFG0 is set. The eUSCI\_B module automatically acknowledges the received data and can receive the next data byte.

If the previous data was not read from the receive buffer UCBxRXBUF at the end of a reception, the bus is stalled by holding SCL low. As soon as UCBxRXBUF is read, the new data is transferred into UCBxRXBUF, an acknowledge is sent to the master, and the next data can be received.

Setting the UCTXNACK bit causes a NACK to be transmitted to the master during the next acknowledgment cycle. A NACK is sent even if UCBxRXBUF is not ready to receive the latest data. If the UCTXNACK bit is set while SCL is held low, the bus is released, a NACK is transmitted immediately, and UCBxRXBUF is loaded with the last received data. Because the previous data was not read, that data is lost. To avoid loss of data, the UCBxRXBUF must be read before UCTXNACK is set.

When the master generates a STOP condition, the UCSTPIFG flag is set.

If the master generates a repeated START condition, the eUSCI\_B I<sup>2</sup>C state machine returns to its address-reception state.

Figure 24-10 shows the I<sup>2</sup>C slave receiver operation.

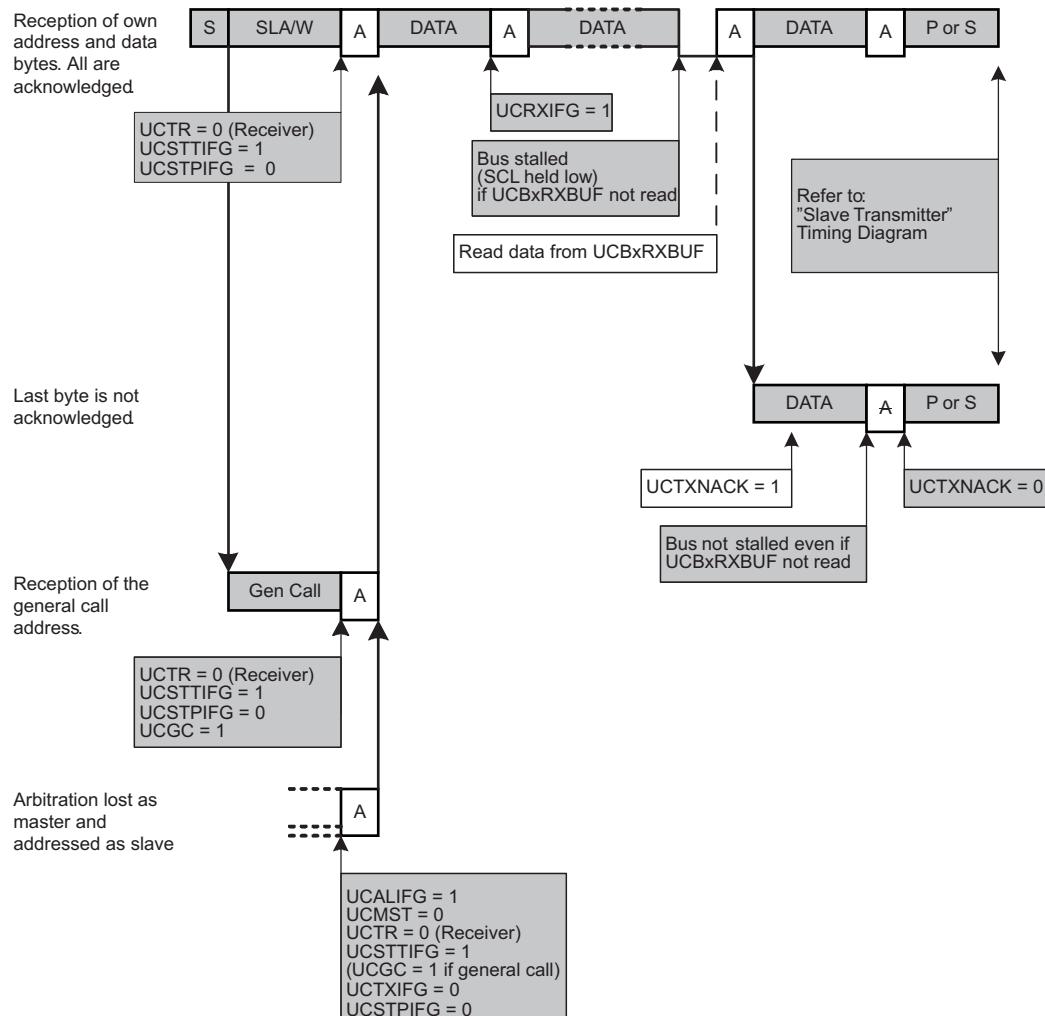
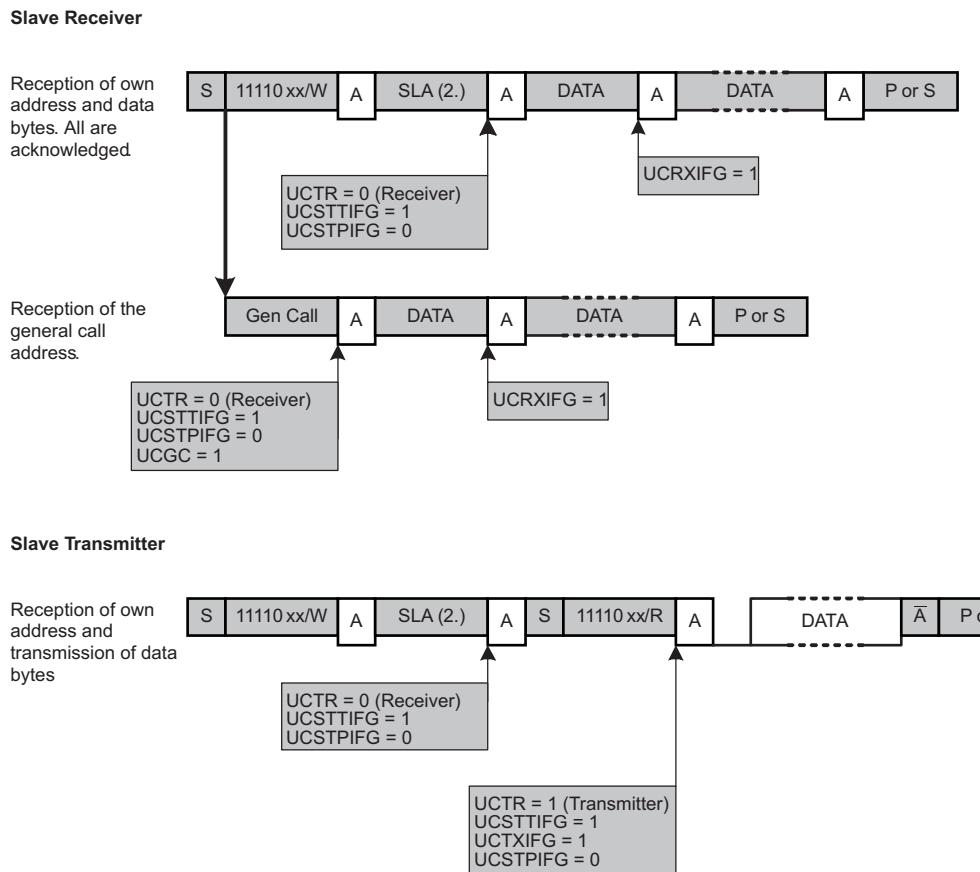


Figure 24-10. I<sup>2</sup>C Slave Receiver Mode

#### 24.3.4.1.3 I<sup>2</sup>C Slave 10-Bit Addressing Mode

The 10-bit addressing mode is selected when UCA10 = 1 (see Figure 24-11). In 10-bit addressing mode, the slave is in receive mode after the full address is received. The eUSCI\_B module indicates this by setting the UCSTTIFG flag while the UCTR bit is cleared. To switch the slave into transmitter mode, the master sends a repeated START condition together with the first byte of the address but with the R/W bit set. This sets the UCSTTIFG flag if it was previously cleared by software, and the eUSCI\_B modules switches to transmitter mode with UCTR = 1.



**Figure 24-11. I<sup>2</sup>C Slave 10-Bit Addressing Mode**

#### 24.3.4.2 Master Mode

The eUSCI\_B module is configured as an I<sup>2</sup>C master by selecting the I<sup>2</sup>C mode with UCMODEEx = 11 and UCSYNC = 1 and setting the UCMST bit. When the master is part of a multi-master system, UCMM must be set and its own address must be programmed into the UCBxI2COA0 register. Support for multiple slave addresses is explained in [Section 24.3.8](#). When UCA10 = 0, 7-bit addressing is selected. When UCA10 = 1, 10-bit addressing is selected. The UCGCEN bit selects if the eUSCI\_B module responds to a general call.

---

**NOTE: Addresses and multi-master systems**

In master mode with own-address detection enabled (UCOAE = 1)—especially in multi-master systems—it is not allowed to specify the same address in the own address and slave address register (UCBxI2CSA = UCBxI2COAx). This would mean that the eUSCI\_B addresses itself.

The user software must ensure that this situation does not occur. There is no hardware detection for this case, and the consequence is unpredictable behavior of the eUSCI\_B.

---

#### 24.3.4.2.1 I<sup>2</sup>C Master Transmitter Mode

After initialization, master transmitter mode is initiated by writing the desired slave address to the UCBxI2CSA register, selecting the size of the slave address with the UCSLA10 bit, setting UCTR for transmitter mode, and setting UCTXSTT to generate a START condition.

The eUSCI\_B module waits until the bus is available, then generates the START condition, and transmits the slave address. The UCTXIFG0 bit is set when the START condition is generated and the first data to be transmitted can be written into UCBxTXBUF. **The UCTXSTT flag is cleared as soon as the complete address is sent.**

The data written into UCBxTXBUF is transmitted if arbitration is not lost during transmission of the slave address. UCTXIFG0 is set again as soon as the data is transferred from the buffer into the shift register. If there is no data loaded to UCBxTXBUF before the acknowledge cycle, the bus is held during the acknowledge cycle with SCL low until data is written into UCBxTXBUF. Data is transmitted or the bus is held, as long as:

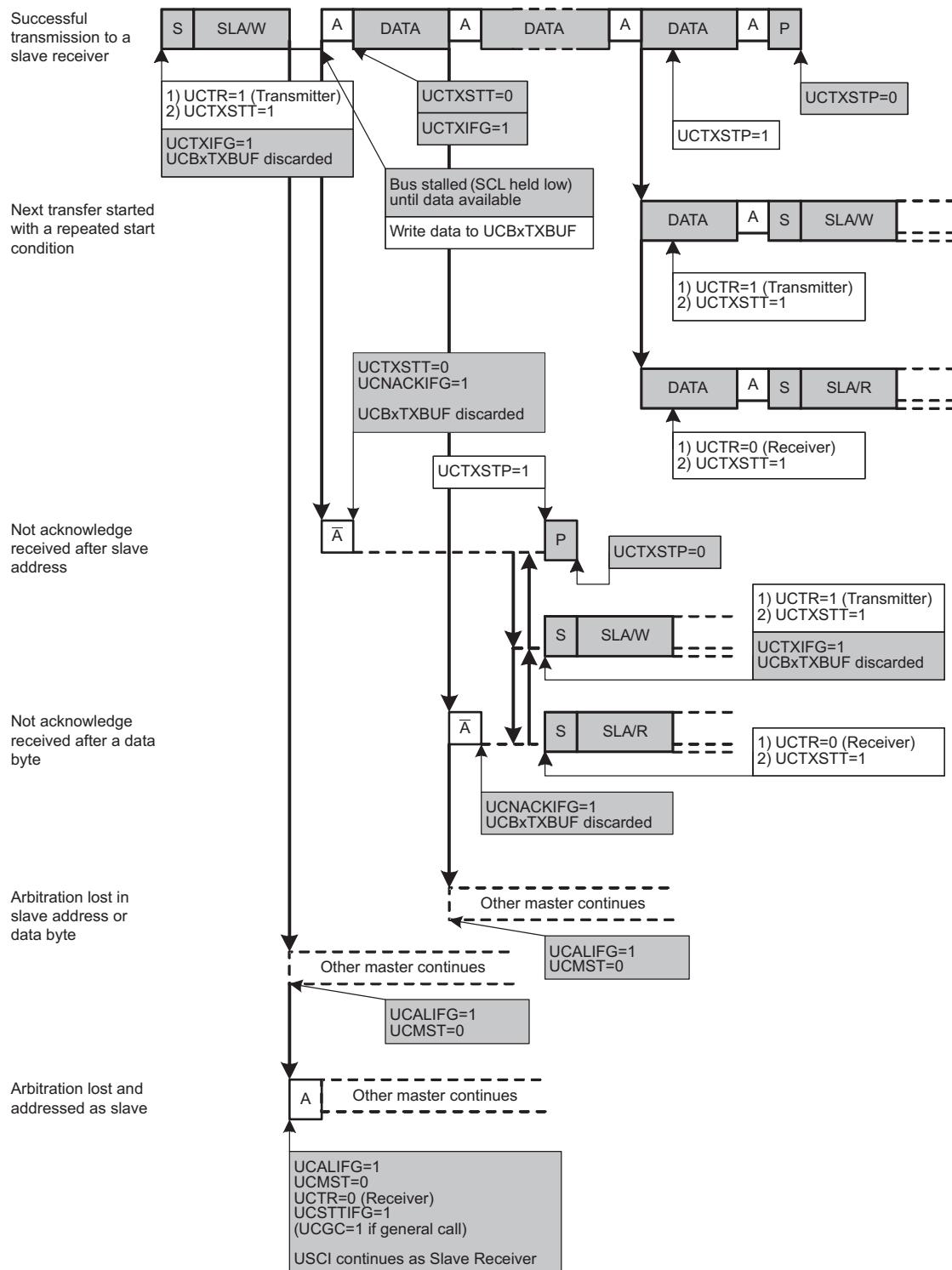
- No automatic STOP is generated
- The UCTXSTP bit is not set
- The UCTXSTT bit is not set

Setting UCTXSTP generates a STOP condition after the next acknowledge from the slave. If UCTXSTP is set during the transmission of the slave address or while the eUSCI\_B module waits for data to be written into UCBxTXBUF, a STOP condition is generated, even if no data was transmitted to the slave. **In this case, the UCSTPIFG is set.** When transmitting a single byte of data, the UCTXSTP bit must be set while the byte is being transmitted or any time after transmission begins, without writing new data into UCBxTXBUF. Otherwise, only the address is transmitted. When the data is transferred from the buffer to the shift register, UCTXIFG0 is set, indicating data transmission has begun, and the UCTXSTP bit may be set. When UCASTPx = 10 is set, the byte counter is used for STOP generation and the user does not need to set the UCTXSTP. **This is recommended when transmitting only one byte.**

Setting UCTXSTT generates a repeated START condition. In this case, UCTR may be set or cleared to configure transmitter or receiver, and a different slave address may be written into UCBxI2CSA, if desired.

If the slave does not acknowledge the transmitted data, the not-acknowledge interrupt flag UCNACKIFG is set. The master must react with either a STOP condition or a repeated START condition. If data was already written into UCBxTXBUF, it is discarded. If this data should be transmitted after a repeated START, it must be written into UCBxTXBUF again. Any set UCTXSTT or UCTXSTP is also discarded.

Figure 24-12 shows the I<sup>2</sup>C master transmitter operation.



**Figure 24-12. I<sup>2</sup>C Master Transmitter Mode**

#### 24.3.4.2.2 I<sup>2</sup>C Master Receiver Mode

After initialization, master receiver mode is initiated by writing the desired slave address to the UCBxI2CSA register, selecting the size of the slave address with the UCSLA10 bit, clearing UCTR for receiver mode, and setting UCTXSTT to generate a START condition.

The eUSCI\_B module checks if the bus is available, generates the START condition, and transmits the slave address. The UCTXSTT flag is cleared as soon as the complete address is sent.

After the acknowledge of the address from the slave, the first data byte from the slave is received and acknowledged and the UCRXIFG flag is set. Data is received from the slave, as long as:

- No automatic STOP is generated
- The UCTXSTP bit is not set
- The UCTXSTT bit is not set

If a STOP condition was generated by the eUSCI\_B module, the UCSTPIFG is set. If UCBxRXBUF is not read, the master holds the bus during reception of the last data bit and until the UCBxRXBUF is read.

If the slave does not acknowledge the transmitted address, the not-acknowledge interrupt flag UCNACKIFG is set. The master must react with either a STOP condition or a repeated START condition.

A STOP condition is either generated by the automatic STOP generation or by setting the UCTXSTP bit. The next byte received from the slave is followed by a NACK and a STOP condition. This NACK occurs immediately if the eUSCI\_B module is currently waiting for UCBxRXBUF to be read.

If a RESTART is sent, UCTR may be set or cleared to configure transmitter or receiver, and a different slave address may be written into UCBxI2CSA if desired.

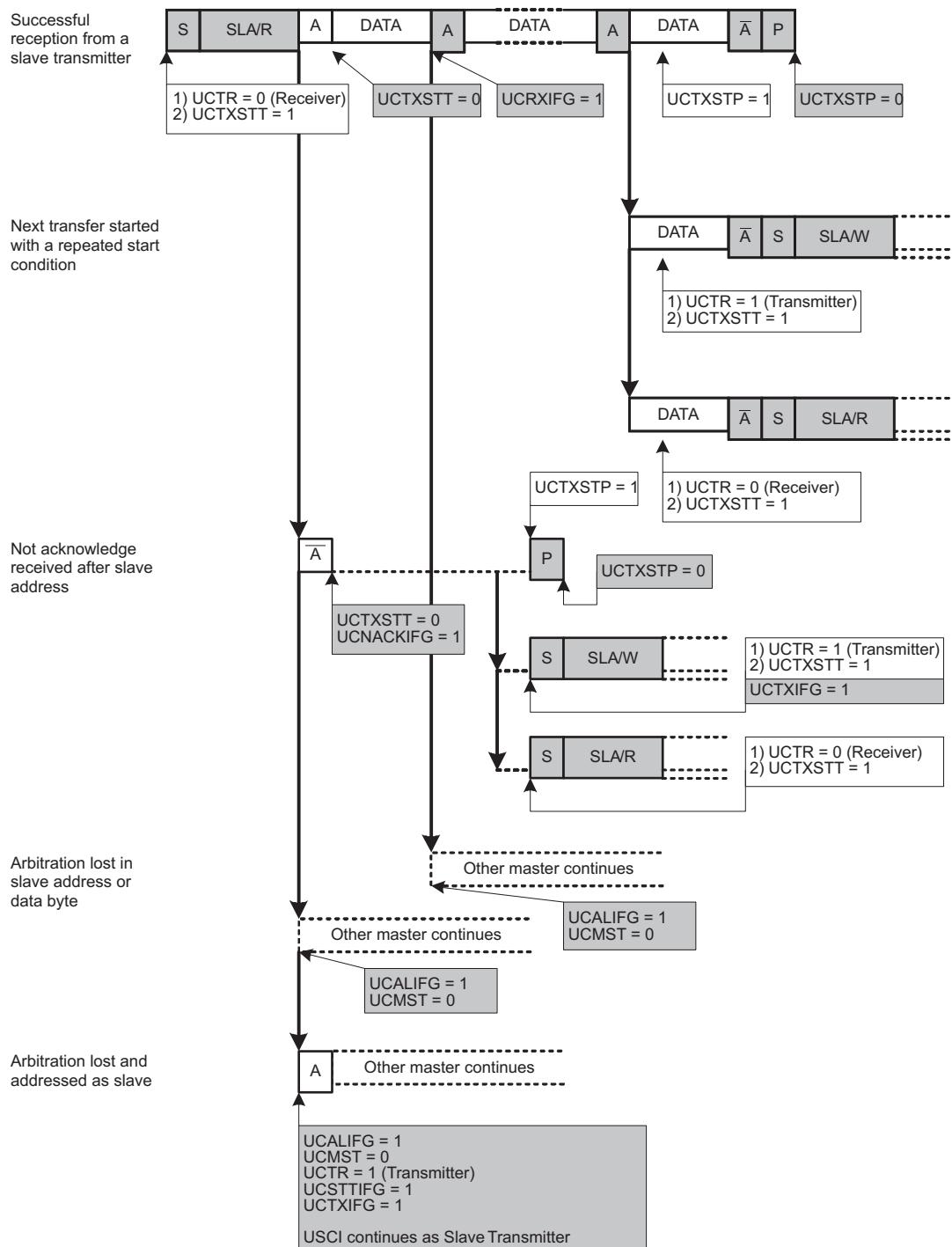
Figure 24-13 shows the I<sup>2</sup>C master receiver operation.

---

**NOTE: Consecutive master transactions without repeated START**

When performing multiple consecutive I<sup>2</sup>C master transactions without the repeated START feature, the current transaction must be completed before the next one is initiated. This can be done by ensuring that the transmit STOP condition flag UCTXSTP is cleared before the next I<sup>2</sup>C transaction is initiated with setting UCTXSTT = 1. Otherwise, the current transaction might be affected.

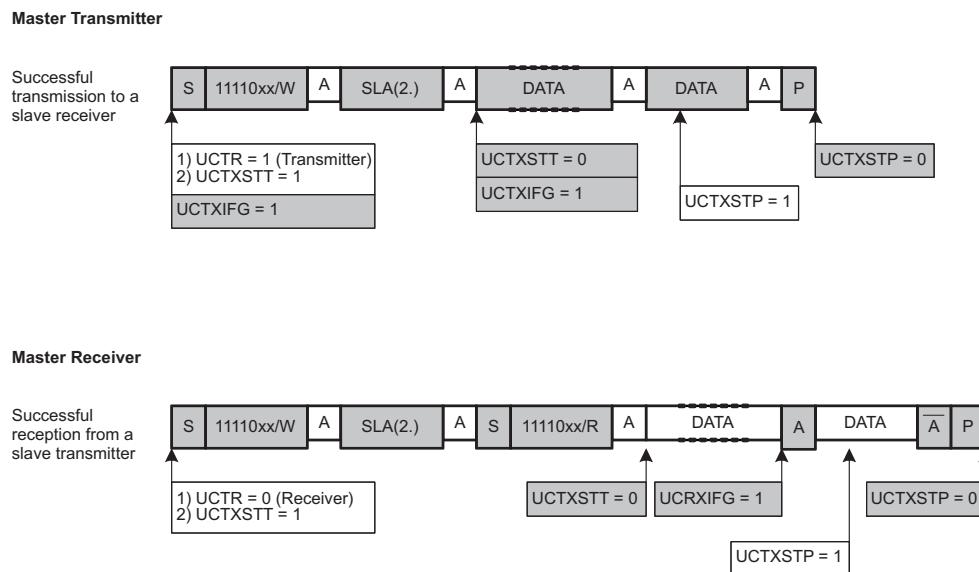
---



**Figure 24-13. I<sup>2</sup>C Master Receiver Mode**

#### 24.3.4.2.3 I<sup>2</sup>C Master 10-Bit Addressing Mode

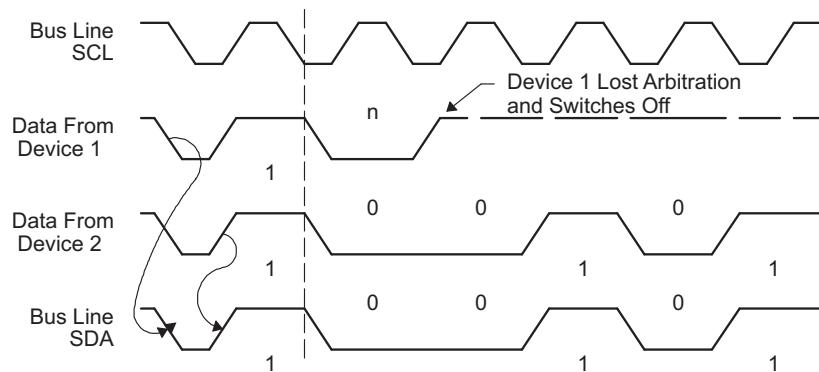
The 10-bit addressing mode is selected when UCSLA10 = 1 (see Figure 24-14).



**Figure 24-14. I<sup>2</sup>C Master 10-Bit Addressing Mode**

#### 24.3.4.3 Arbitration

If two or more master transmitters simultaneously start a transmission on the bus, an arbitration procedure is invoked. Figure 24-15 shows the arbitration procedure between two devices. The arbitration procedure uses the data presented on SDA by the competing transmitters. The first master transmitter that generates a logic high is overruled by the opposing master generating a logic low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. The master transmitter that lost arbitration switches to the slave receiver mode and sets the arbitration lost flag UCALIFG. If two or more devices send identical first bytes, arbitration continues on the subsequent bytes.



**Figure 24-15. Arbitration Procedure Between Two Master Transmitters**

There is an undefined condition if the arbitration procedure is still in progress when one master sends a repeated START or a STOP condition while the other master is still sending data. In other words, the following combinations result in an undefined condition:

- Master 1 sends a repeated START condition and master 2 sends a data bit.
- Master 1 sends a STOP condition and master 2 sends a data bit.
- Master 1 sends a repeated START condition and master 2 sends a STOP condition.

### 24.3.5 Glitch Filtering

According to the I<sup>2</sup>C standard, both the SDA and the SCL line need to be glitch filtered. The eUSCI\_B module provides the UCGLITx bits to configure the length of this glitch filter (see [Table 24-1](#)).

**Table 24-1. Glitch Filter Length Selection Bits**

UCGLITx	Corresponding Glitch Filter Length on SDA and SCL	According to I <sup>2</sup> C Standard
00	Pulses of maximum 50-ns length are filtered.	yes
01	Pulses of maximum 25-ns length are filtered.	no
10	Pulses of maximum 12.5-ns length are filtered.	no
11	Pulses of maximum 6.25-ns length are filtered.	no

### 24.3.6 I<sup>2</sup>C Clock Generation and Synchronization

The I<sup>2</sup>C clock SCL is provided by the master on the I<sup>2</sup>C bus. When the eUSCI\_B is in master mode, BITCLK is provided by the eUSCI\_B bit clock generator and the clock source is selected with the UCSSELx bits. In slave mode, the bit clock generator is not used and the UCSSELx bits are don't care.

The 16-bit value of UCBRx in register UCBxBRW is the division factor of the eUSCI\_B clock source, BRCLK. The maximum bit clock that can be used in single master mode is  $f_{BRCLK}/4$ . In multi-master mode, the maximum bit clock is  $f_{BRCLK}/8$ . The BITCLK frequency is given by:

$$f_{BitClock} = f_{BRCLK}/UCBRx$$

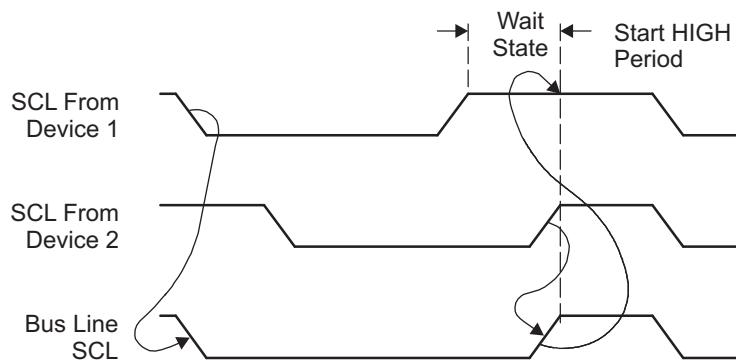
The minimum high and low periods of the generated SCL are:

$$t_{LOW,MIN} = t_{HIGH,MIN} = (UCBRx/2)/f_{BRCLK} \text{ when UCBRx is even}$$

$$t_{LOW,MIN} = t_{HIGH,MIN} = ((UCBRx - 1)/2)/f_{BRCLK} \text{ when UCBRx is odd}$$

The eUSCI\_B clock source frequency and the prescaler setting UCBRx must to be chosen such that the minimum low and high period times of the I<sup>2</sup>C specification are met.

During the arbitration procedure the clocks from the different masters must be synchronized. A device that first generates a low period on SCL overrules the other devices, forcing them to start their own low periods. SCL is then held low by the device with the longest low period. The other devices must wait for SCL to be released before starting their high periods. [Figure 24-16](#) shows the clock synchronization. This allows a slow slave to slow down a fast master.



**Figure 24-16. Synchronization of Two I<sup>2</sup>C Clock Generators During Arbitration**

#### 24.3.6.1 Clock Stretching

The eUSCI\_B module supports clock stretching and also makes use of this feature as described in the Operation Mode sections.

The UCSCLLOW bit can be used to observe if another device pulls SCL low while the eUSCI\_B module already released SCL due to the following conditions:

- eUSCI\_B is acting as master and a connected slave drives SCL low.

- eUSCI\_B is acting as master and another master drives SCL low during arbitration.

The UCSCLLOW bit is also active if the eUSCI\_B holds SCL low because it is waiting as transmitter for data being written into UCBxTXBUF or as receiver for the data being read from UCBxRXBUF. The UCSCLLOW bit might be set for a short time with each rising SCL edge because the logic observes the external SCL and compares it to the internally generated SCL.

#### 24.3.6.2 Avoiding Clock Stretching

Even though clock stretching is part of the I<sup>2</sup>C specification, there are applications in which clock stretching should be avoided.

The clock is stretched by the eUSCI\_B under the following conditions:

- The internal shift register is expecting data, but the TXIFG is still pending
- The internal shift register is full, but the RXIFG is still pending
- The arbitration lost interrupt is pending
- UCSWACK is selected and UCBxI2COA0 did cause a match

To avoid clock stretching, all of these situations for clock stretch either need to be avoided or the corresponding interrupt flags need to be processed before the actual clock stretch can occur.

Using the DMA (on devices that contain a DMA) is the most secure way to avoid clock stretching. If no DMA is available, the software must ensure that the corresponding interrupts are serviced in time before the clock is stretched.

In slave transmitter mode, the TXIFG is set only after the reception of the direction bit; therefore, there is only a short amount of time for the software to write the TXBUF before a clock stretch occurs. This situation can be remedied by using the early Transmit Interrupt (see [Section 24.3.10.2](#)).

#### 24.3.6.3 Clock Low Timeout

The UCCLTOIFG interrupt allows the software to react if the clock is low longer than a defined time. It is possible to detect the situation, when a clock is stretched by a master or slave for a too long time. The user can then, for example, reset the eUSCI\_B module by using the UCSWRST bit.

The clock low time-out feature is enabled using the UCCLTO bits. It is possible to select one of three predefined times for the clock low time-out. If the clock has been low longer than the time defined with the UCCLTO bits and the eUSCI\_B was actively receiving or transmitting, the UCCLTOIFG is set and an interrupt request is generated if UCCLTOIE is set as well. The UCCLTOIFG is set only once, even if the clock is stretched a multiple of the time defined in UCCLTO.

### 24.3.7 Byte Counter

The eUSCI\_B module supports hardware counting of the bytes received or transmitted. The counter is automatically active and counts up for each byte seen on the bus in both master and slave mode.

The byte counter is incremented at the second bit position of each byte independently of the following ACK or NACK. A START or RESTART condition resets the counter value to zero. Address bytes do not increment the counter. The byte counter is also incremented at the second bit position, if an arbitration loss occurs during the first bit of data.

#### 24.3.7.1 Byte Counter Interrupt

If UCASTPx = 01 or 10 the UCBCNTIFG is set when the byte counter threshold value UCBxTBCNT is reached in both master- and slave-mode. Writing zero to UCBxTBCNT does not generate an interrupt.

Because the UCBCNTIFG has a lower interrupt priority than the UCBTXIFG and UCBRXIFG, TI recommends using it only for protocol control together with the DMA handling the received and transmitted bytes. Otherwise, the application must have enough processor bandwidth to ensure that the UCBCNT interrupt routine is executed in time to generate, for example, a RESTART.

#### 24.3.7.2 Automatic STOP Generation

When the eUSCI\_B module is configured as a master, the byte counter can be used for automatic STOP generation by setting the UCASTPx = 10. Before starting the transmission using UCTXSTT, the byte counter threshold UCBxTBCNT must be set to the number of bytes that are to be transmitted or received. After the number of bytes that are configured in UCBxTBCNT have been transmitted, the eUSCI\_B automatically generates a STOP condition.

UCBxTBCNT cannot be used if the user wants to transmit the slave address only without any data. In this case, TI recommends setting UCTXSTT and UCTXSTP at the same time.

### 24.3.8 Multiple Slave Addresses

The eUSCI\_B module supports two different ways of implementing multiple slave addresses at the same time:

- Hardware support for up to 4 different slave addresses, each with its own interrupt flag and DMA trigger
- Software support for up to  $2^{10}$  different slave addresses all sharing one interrupt

#### 24.3.8.1 Multiple Slave Address Registers

The registers UCBxI2COA0, UCBxI2COA1, UCBxI2COA2, and UCBxI2COA3 contain four slave addresses. Up to four address registers are compared against a received 7- or 10-bit address. Each slave address must be activated by setting the UCOAEN bit in the corresponding UCBxI2COAx register. Register UCBxI2COA3 has the highest priority if the address received on the bus matches more than one of the slave address registers. The priority decreases with the index number of the address register, so that UCBxI2COA0 in combination with the address mask has the lowest priority.

When one of the slave registers matches the 7- or 10-bit address seen on the bus, the address is acknowledged. In the following the corresponding receive- or transmit-interrupt flag (UCTXIFGx or UCRXIFGx) to the received address is updated. The state change interrupt flags are independent of the address comparison result. They are updated according to the bus condition.

#### 24.3.8.2 Address Mask Register

The Address Mask Register can be used when the eUSCI\_B is configured in slave or in multiple-master mode. To activate this feature, at least one bit of the address mask in register UCBxADDMASK must be cleared.

If the received address matches the own address in UCBxI2COA0 on all bit positions not masked by UCBxADDMASK the eUSCI\_B considers the seen address as its own address and sends an acknowledge. The user has the choice to either automatically acknowledge the address seen on the bus or to evaluate this address and send the acknowledge in software using UCTXACK. The selection between these options is done using the UCSWACK bit. If the software is used for generation of the ACK of the slave address, TI recommends using the UCSTTIFG. The received address can be found in the UCBxADDRX register.

A slave address seen on the bus is automatically acknowledged by the eUSCI\_B module, if it matches any of the slave addresses defined in UCBxI2COA1 to UCBxI2COA3.

---

**NOTE: UCSWACK and slave-transmitter**

If the user selects manual acknowledge of slave addresses, the TXIFG is set if the slave is addressed as a transmitter. If the user decides not to acknowledge the address, the TXIFG also must be reset.

---

#### 24.3.9 Using the eUSCI\_B Module in I<sup>2</sup>C Mode With Low-Power Modes

The eUSCI module is not functional when the device is in LPM3, LPM4, or LPMx.5 modes of operation. However, the application can make use of the FORCE\_LPM\_ENTRY bit in the PCMCTL1 register to determine whether the entry to low-power mode should be aborted if the eUSCI is active, or if the device can continue low power entry regardless. The latter option is useful if the eUSCI is transmitting and receiving data at a very slow rate, and the application can tolerate entry to low-power mode at the expense of a packet of data being lost. Refer to the *Power Control Manager (PCM)* chapter for more details.

#### 24.3.10 eUSCI\_B Interrupts in I<sup>2</sup>C Mode

The eUSCI\_B has only one interrupt vector that is shared for transmission, reception, and the state change.

Each interrupt flag has its own interrupt enable bit. When an interrupt is enabled, the interrupt flag generates an interrupt request. DMA transfers are controlled by the UCTXIFGx and UCRXIFGx flags on devices with a DMA controller. It is possible to react on each slave address with an individual DMA channel.

All interrupt flags are not cleared automatically, but they need to be cleared together by user interactions (for example, reading the UCRXB0F clears UCRXIFGx). If the user wants to use an interrupt flag he needs to ensure that the flag has the correct state before the corresponding interrupt is enabled.

#### 24.3.10.1 I<sup>2</sup>C Transmit Interrupt Operation

The UCTXIFG0 interrupt flag is set whenever the transmitter is able to accept a new byte. When operating as a slave with multiple slave addresses, the UCTXIFGx flags are set corresponding to which address was received before. If, for example, the slave address specified in register UCBxI2COA3 did match the address seen on the bus, the UCTXIFG3 indicates that the UCBxTXBUF is ready to accept a new byte.

When operating in master mode with automatic STOP generation (UCASTPx = 10), the UCTXIFG0 is set as many times as defined in UCBxTBCNT.

An interrupt request is generated if UCTXIE is set. UCTXIFGx is automatically reset if a write to UCBxTXBUF occurs or if the UCALIFG is cleared. UCTXIFGx is set when:

- Master mode: UCTXSTT was set by the user
- Slave mode: own address was received(UCETXINT = 0) or START was received (UCETXINT = 1)

UCTXIE is reset after a Hard Reset or when UCSWRST = 1.

#### 24.3.10.2 Early I<sup>2</sup>C Transmit Interrupt

Setting the UCETXINT causes UCTXIFG0 to be sent out automatically when a START condition is sent and the eUSCI\_B is configured as slave. In this case, it is not allowed to enable the other slave addresses UCBxI2COA1-UCBxI2COA3. This allows the software more time to handle the UCTXIFG0 compared to the normal situation, when UCTXIFG0 is sent out after the slave address match was detected. Situations where the UCTXIFG0 was set and afterward no slave address match occurred need to be handled in software. TI recommends using the byte counter to handle this.

#### 24.3.10.3 I<sup>2</sup>C Receive Interrupt Operation

The UCRXIFG0 interrupt flag is set when a character is received and loaded into UCBxRXBUF. When operating as a slave with multiple slave addresses, the UCRXIFGx flag is set corresponding to which address was received before.

An interrupt request is generated if UCRXIE is set. UCRXIFGx and UCRXIE are reset after a Hard Reset signal or when UCSWRST = 1. UCRXIFGx is automatically reset when UCxRXBUF is read.

#### 24.3.10.4 I<sup>2</sup>C State Change Interrupt Operation

Table 24-2 describes the I<sup>2</sup>C state change interrupt flags.

**Table 24-2. I<sup>2</sup>C State Change Interrupt Flags**

Interrupt Flag	Interrupt Condition
UCALIFG	Arbitration-lost. Arbitration can be lost when two or more transmitters start a transmission simultaneously, or when the eUSCI_B operates as master but is addressed as a slave by another master in the system. The UCALIFG flag is set when arbitration is lost. When UCALIFG is set, the UCMST bit is cleared and the I <sup>2</sup> C controller becomes a slave.
UCNACKIFG	Not-acknowledge interrupt. This flag is set when an acknowledge is expected but is not received. UCNACKIFG is used in master mode only.
UCCLTOIFG	Clock low time-out. This interrupt flag is set, if the clock is held low longer than defined by the UCCLTO bits.
UCBIT9IFG	This interrupt flag is generated each time the eUSCI_B is transferring 9th clock cycle of a byte of data. This gives the user the possibility to follow the I <sup>2</sup> C communication in software if wanted. The UCBIT9IFG is not set for address information.
UCBCNTIFG	Byte counter interrupt. This flag is set when the byte counter value reaches the value defined in UCBxTBCNT and UCASPx = 01 or 10. This bit allows to organize following communications, especially if a RESTART will be issued.
UCSTTIIFG	START condition detected interrupt. This flag is set when the I <sup>2</sup> C module detects a START condition together with its own address <sup>(1)</sup> . UCSTTIIFG is used in slave mode only.
UCSTPIFG	STOP condition detected interrupt. This flag is set when the I <sup>2</sup> C module detects a STOP condition on the bus. UCSTPIFG is used in slave and master mode.

<sup>(1)</sup> The address evaluation includes the address mask register if it is used.

#### 24.3.10.5 UCBxIV, Interrupt Vector Generator

The eUSCI\_B interrupt flags are prioritized and combined to source a single interrupt vector. The interrupt vector register UCBxIV is used to determine which flag requested an interrupt. The highest-priority enabled interrupt generates a number in the UCBxIV register that can be evaluated or added to the PC to automatically enter the appropriate software routine. Disabled interrupts do not affect the UCBxIV value.

Read access of the UCBxIV register automatically resets the highest-pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt.

Write access of the UCBxIV register clears all pending Interrupt conditions and flags.

## 24.4 eUSCI\_B I<sup>2</sup>C Registers

The eUSCI\_B registers applicable in I<sup>2</sup>C mode and their address offsets are listed in [Table 24-3](#). The base address can be found in the device-specific data sheet.

**Table 24-3. eUSCI\_B Registers**

Offset	Acronym	Register Name	Section
00h	UCBxCTLW0	eUSCI_Bx Control Word 0	<a href="#">Section 24.4.1</a>
00h	UCBxCTL1	eUSCI_Bx Control 1	
01h	UCBxCTL0	eUSCI_Bx Control 0	
02h	UCBxCTLW1	eUSCI_Bx Control Word 1	<a href="#">Section 24.4.2</a>
06h	UCBxBRW	eUSCI_Bx Bit Rate Control Word	<a href="#">Section 24.4.3</a>
06h	UCBxBR0	eUSCI_Bx Bit Rate Control 0	
07h	UCBxBR1	eUSCI_Bx Bit Rate Control 1	
08h	UCBxSTATW	eUSCI_Bx Status Word	<a href="#">Section 24.4.4</a>
08h	UCBxSTAT	eUSCI_Bx Status	
09h	UCBxBCNT	eUSCI_Bx Byte Counter	
0Ah	UCBxTBCNT	eUSCI_Bx Byte Counter Threshold	<a href="#">Section 24.4.5</a>
0Ch	UCBxRXBUF	eUSCI_Bx Receive Buffer	<a href="#">Section 24.4.6</a>
0Eh	UCBxTXBUF	eUSCI_Bx Transmit Buffer	<a href="#">Section 24.4.7</a>
14h	UCBxI2COA0	eUSCI_Bx I <sup>2</sup> C Own Address 0	<a href="#">Section 24.4.8</a>
16h	UCBxI2COA1	eUSCI_Bx I <sup>2</sup> C Own Address 1	<a href="#">Section 24.4.9</a>
18h	UCBxI2COA2	eUSCI_Bx I <sup>2</sup> C Own Address 2	<a href="#">Section 24.4.10</a>
1Ah	UCBxI2COA3	eUSCI_Bx I <sup>2</sup> C Own Address 3	<a href="#">Section 24.4.11</a>
1Ch	UCBxADDRX	eUSCI_Bx Received Address	<a href="#">Section 24.4.12</a>
1Eh	UCBxADDMASK	eUSCI_Bx Address Mask	<a href="#">Section 24.4.13</a>
20h	UCBxI2CSA	eUSCI_Bx I <sup>2</sup> C Slave Address	<a href="#">Section 24.4.14</a>
2Ah	UCBxIE	eUSCI_Bx Interrupt Enable	<a href="#">Section 24.4.15</a>
2Ch	UCBxIFG	eUSCI_Bx Interrupt Flag	<a href="#">Section 24.4.16</a>
2Eh	UCBxIV	eUSCI_Bx Interrupt Vector	<a href="#">Section 24.4.17</a>

---

**NOTE:** This is a 16-bit module and must be accessed ONLY through byte (8 bit) or half-word (16 bit) access. 32-bit read or write access to this module causes a bus error.

---

For details on the register bit access and reset conventions that are used in the following sections, refer to [Preface](#).

#### 24.4.1 UCBxCTLW0 Register

eUSCI\_Bx Control Word Register 0

**Figure 24-17. UCBxCTLW0 Register**

15	14	13	12	11	10	9	8
UCA10	UCSLA10	UCMM	Reserved	UCMST	UCMODEx	UCSYNC	
rw-0	rw-0	rw-0	r0	rw-0	rw-0	rw-0	r1
7	6	5	4	3	2	1	0
UCSELx	UCTXACK	UCTR	UCTXNACK	UCTXSTP	UCTXSTT	UCSWRST	
rw-1	rw-1	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1
Modify only when UCSWRST = 1.							

**Table 24-4. UCBxCTLW0 Register Description**

Bit	Field	Type	Reset	Description
15	UCA10	RW	0h	Own addressing mode select. Modify only when UCSWRST = 1. 0b = Own address is a 7-bit address. 1b = Own address is a 10-bit address.
14	UCSLA10	RW	0h	Slave addressing mode select 0b = Address slave with 7-bit address 1b = Address slave with 10-bit address
13	UCMM	RW	0h	Multi-master environment select. Modify only when UCSWRST = 1. 0b = Single master environment. There is no other master in the system. The address compare unit is disabled. 1b = Multi-master environment
12	Reserved	R	0h	Reserved
11	UCMST	RW	0h	Master mode select. When a master loses arbitration in a multi-master environment (UCMM = 1), the UCMST bit is automatically cleared and the module acts as slave. 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI_B mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. Modify only when UCSWRST = 1. 00b = 3-pin SPI 01b = 4-pin SPI (master or slave enabled if STE = 1) 10b = 4-pin SPI (master or slave enabled if STE = 0) 11b = I <sup>2</sup> C mode
8	UCSYNC	RW	1h	Synchronous mode enable. For eUSCI_B always read and write as 1.
7-6	UCSELx	RW	3h	eUSCI_B clock source select. These bits select the BRCLK source clock. These bits are ignored in slave mode. Modify only when UCSWRST = 1. 00b = UCLKI 01b = ACLK 10b = SMCLK 11b = SMCLK
5	UCTXACK	RW	0h	Transmit ACK condition in slave mode with enabled address mask register. After the UCSTTIFG has been set, the user needs to set or reset the UCTXACK flag to continue with the I <sup>2</sup> C protocol. The clock is stretched until the UCBxCTL1 register has been written. This bit is cleared automatically after the ACK has been send. 0b = Do not acknowledge the slave address 1b = Acknowledge the slave address

**Table 24-4. UCBxCTLW0 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
4	UCTR	RW	0h	Transmitter/receiver 0b = Receiver 1b = Transmitter
3	UCTXNACK	RW	0h	Transmit a NACK. UCTXNACK is automatically cleared after a NACK is transmitted. Only for slave receiver mode. 0b = Acknowledge normally 1b = Generate NACK
2	UCTXSTP	RW	0h	Transmit STOP condition in master mode. Ignored in slave mode. In master receiver mode, the STOP condition is preceded by a NACK. UCTXSTP is automatically cleared after STOP is generated. This bit is a don't care, if automatic UCASPx is different from 01 or 10. 0b = No STOP generated 1b = Generate STOP
1	UCTXSTT	RW	0h	Transmit START condition in master mode. Ignored in slave mode. In master receiver mode, a repeated START condition is preceded by a NACK. UCTXSTT is automatically cleared after START condition and address information is transmitted. Ignored in slave mode. 0b = Do not generate START condition 1b = Generate START condition
0	UCSWRST	RW	1h	Software reset enable. 0b = Disabled. eUSCI_B released for operation. 1b = Enabled. eUSCI_B logic held in reset state.

#### 24.4.2 UCBxCTLW1 Register

eUSCI\_Bx Control Word Register 1

**Figure 24-18. UCBxCTLW1 Register**

15	14	13	12	11	10	9	8
Reserved							UCETXINT
r0	r0	r0	r0	r0	r0	r0	rw-0
7	6	5	4	3	2	1	0
UCCLTO		UCSTPNACK	UCSWACK	UCASTPx		UCGLITx	
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
Modify only when UCSWRST = 1.							

**Table 24-5. UCBxCTLW1 Register Description**

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved
8	UCETXINT	RW	0h	Early UCTXIFG0. Only in slave mode. When this bit is set, the slave addresses defined in UCxI2COA1 to UCxI2COA3 must be disabled. Modify only when UCSWRST = 1. 0b = UCTXIFGx is set after an address match with UCxI2COAx and the direction bit indicating slave transmit 1b = UCTXIFG0 is set for each START condition
7-6	UCCLTO	RW	0h	Clock low timeout select. Modify only when UCSWRST = 1. 00b = Disable clock low timeout counter 01b = 135 000 SYSCLK cycles (approximately 28 ms) 10b = 150 000 SYSCLK cycles (approximately 31 ms) 11b = 165 000 SYSCLK cycles (approximately 34 ms)
5	UCSTPNACK	RW	0h	The UCSTPNACK bit allows to make the eUSCI_B master acknowledge the last byte in master receiver mode as well. This is not conform to the I <sup>2</sup> C specification and should only be used for slaves, which automatically release the SDA after a fixed packet length. Modify only when UCSWRST = 1. 0b = Send a not acknowledge before the STOP condition as a master receiver (conform to I <sup>2</sup> C standard) 1b = All bytes are acknowledged by the eUSCI_B when configured as master receiver
4	UCSWACK	RW	0h	Using this bit it is possible to select, whether the eUSCI_B module triggers the sending of the ACK of the address or if it is controlled by software. 0b = The address acknowledge of the slave is controlled by the eUSCI_B module 1b = The user needs to trigger the sending of the address ACK by issuing UCTXACK
3-2	UCASTPx	RW	0h	Automatic STOP condition generation. In slave mode only UCBCNTIFG is available. Modify only when UCSWRST = 1. 00b = No automatic STOP generation. The STOP condition is generated after the user sets the UCTXSTP bit. The value in UCBxTBCNT is a don't care. 01b = UCBCNTIFG is set with the byte counter reaches the threshold defined in UCBxTBCNT 10b = A STOP condition is generated automatically after the byte counter value reached UCBxTBCNT. UCBCNTIFG is set with the byte counter reaching the threshold. 11b = Reserved

**Table 24-5. UCBxCTLW1 Register Description (continued)**

<b>Bit</b>	<b>Field</b>	<b>Type</b>	<b>Reset</b>	<b>Description</b>
1-0	UCGLITx	RW	0h	Deglitch time 00b = 50 ns 01b = 25 ns 10b = 12.5 ns 11b = 6.25 ns

### 24.4.3 UCBxBRW Register

eUSCI\_Bx Bit Rate Control Word Register

**Figure 24-19. UCBxBRW Register**

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
Modify only when UCSWRST = 1.							

**Table 24-6. UCBxBRW Register Description**

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler. Modify only when UCSWRST = 1.

### 24.4.4 UCBxSTATW

eUSCI\_Bx Status Word Register

**Figure 24-20. UCBxSTATW Register**

15	14	13	12	11	10	9	8
UCBCNTx							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
7	6	5	4	3	2	1	0
Reserved	UCSCLLOW	UCGC	UCBBUSY	Reserved			
r0	r-0	r-0	r-0	r-0	r0	r0	r0

**Table 24-7. UCBxSTATW Register Description**

Bit	Field	Type	Reset	Description
15-8	UCBCNTx	R	0h	Hardware byte counter value. Reading this register returns the number of bytes received or transmitted on the I <sup>2</sup> C bus since the last START or RESTART. There is no synchronization of this register done. When reading UCBxBCNT during the first bit position, a faulty read can occur.
7	Reserved	R	0h	Reserved
6	UCSCLLOW	R	0h	SCL low 0b = SCL is not held low 1b = SCL is held low
5	UCGC	R	0h	General call address received. UCGC is automatically cleared when a START condition is received. 0b = No general call address received 1b = General call address received
4	UCBBUSY	R	0h	Bus busy 0b = Bus inactive 1b = Bus busy
3-0	Reserved	R	0h	Reserved

#### **24.4.5 UCBxTBCNT Register**

eUSCI\_Bx Byte Counter Threshold Register

**Figure 24-21. UCBxTBCNT Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTBCNTx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
Modify only when UCSWRST = 1.							

**Table 24-8. UCBxTBCNT Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTBCNTx	RW	0h	The byte counter threshold value is used to set the number of I <sup>2</sup> C data bytes after which the automatic STOP or the UCSTPIFG should occur. This value is evaluated only if UCASTPx is different from 00. Modify only when UCSWRST = 1.

#### 24.4.6 UCBxRXBUF Register

eUSCI\_Bx Receive Buffer Register

**Figure 24-22. UCBxRXBUF Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCRXBUFx							
r	r	r	r	r	r	r	r

**Table 24-9. UCBxRXBUF Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCRXBUFx	R	0h	The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCBxRXBUF resets the UCRXIFGx flags.

#### 24.4.7 UCBxTXBUF

eUSCI\_Bx Transmit Buffer Register

**Figure 24-23. UCBxTXBUF Register**

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCTXBUFx							
rw	rw	rw	rw	rw	rw	rw	rw

**Table 24-10. UCBxTXBUF Register Description**

Bit	Field	Type	Reset	Description
15-8	Reserved	R	0h	Reserved
7-0	UCTXBUFx	RW	0h	The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears the UCTXIFGx flags.

#### **24.4.8 UCBxI2COA0 Register**

eUSCI\_Bx I<sup>2</sup>C Own Address 0 Register

**Figure 24-24. UCBxI2COA0 Register**

15	14	13	12	11	10	9	8
UCGCEN	Reserved				UCOAEN	I2COA0	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA0							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
Modify only when UCSWRST = 1.							

**Table 24-11. UCBxI2COA0 Register Description**

Bit	Field	Type	Reset	Description
15	UCGCEN	RW	0h	General call response enable. This bit is only available in UCBxI2COA0. Modify only when UCSWRST = 1. 0b = Do not respond to a general call 1b = Respond to a general call
14-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I <sup>2</sup> C slave-address related to this register UCBxI2COA0 is evaluated or not. Modify only when UCSWRST = 1. 0b = The slave address defined in I2COA0 is disabled 1b = The slave address defined in I2COA0 is enabled
9-0	I2COA0	RW	0h	I <sup>2</sup> C own address. The I2COA0 bits contain the local address of the eUSCI <sub>x</sub> _B I <sup>2</sup> C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Modify only when UCSWRST = 1.

#### 24.4.9 UCBxI2COA1 Register

eUSCI\_Bx I2C Own Address 1 Register

**Figure 24-25. UCBxI2COA1 Register**

15	14	13	12	11	10	9	8
Reserved				UCOAEN		I2COA1	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA1							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
Modify only when UCSWRST = 1.							

**Table 24-12. UCBxI2COA1 Register Description**

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA1 is evaluated or not. Modify only when UCSWRST = 1. 0b = The slave address defined in I2COA1 is disabled 1b = The slave address defined in I2COA1 is enabled
9-0	I2COA1	RW	0h	I2C own address. The I2COA1 bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Modify only when UCSWRST = 1.

#### 24.4.10 UCBxI2COA2 Register

eUSCI\_Bx I2C Own Address 2 Register

**Figure 24-26. UCBxI2COA2 Register**

15	14	13	12	11	10	9	8
Reserved				UCOAEN		I2COA2	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA2							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
Modify only when UCSWRST = 1.							

**Table 24-13. UCBxI2COA2 Register Description**

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	UCOAEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA2 is evaluated or not. Modify only when UCSWRST = 1. 0b = The slave address defined in I2COA2 is disabled 1b = The slave address defined in I2COA2 is enabled
9-0	I2COA2	RW	0h	I2C own address. The I2COA2 bits contain the local address of the eUSCIx_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Modify only when UCSWRST = 1.

#### **24.4.11 UCBxI2COA3 Register**

eUSCI\_Bx I2C Own Address 3 Register

**Figure 24-27. UCBxI2COA3 Register**

15	14	13	12	11	10	9	8
Reserved				UCOEN		I2COA3	
rw-0	r0	r0	r0	r0	rw-0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2COA3							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0
Modify only when UCSWRST = 1.							

**Table 24-14. UCBxI2COA3 Register Description**

Bit	Field	Type	Reset	Description
15-11	Reserved	R	0h	Reserved
10	UCOEN	RW	0h	Own Address enable register. With this register it can be selected if the I2C slave-address related to this register UCBxI2COA3 is evaluated or not. Modify only when UCSWRST = 1. 0b = The slave address defined in I2COA3 is disabled 1b = The slave address defined in I2COA3 is enabled
9-0	I2COA3	RW	0h	I2C own address. The I2COA3 bits contain the local address of the eUSCI_B I2C controller. The address is right justified. In 7-bit addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB. Modify only when UCSWRST = 1.

#### **24.4.12 UCBxADDRX Register**

eUSCI\_Bx I2C Received Address Register

**Figure 24-28. UCBxADDRX Register**

15	14	13	12	11	10	9	8
Reserved							ADDRXX
r-0	r0	r0	r0	r0	r0	r-0	r-0
7	6	5	4	3	2	1	0
ADDRXX							
r-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0

**Table 24-15. UCBxADDRX Register Description**

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ADDRXX	R	0h	Received Address Register. This register contains the last received slave address on the bus. Using this register and the address mask register it is possible to react on more than one slave address using one eUSCI_B module.

#### 24.4.13 UCBxADDMASK Register

eUSCI\_Bx I<sup>2</sup>C Address Mask Register

**Figure 24-29. UCBxADDMASK Register**

15	14	13	12	11	10	9	8
Reserved						ADDMASKx	
r-0	r0	r0	r0	r0	r0	rw-1	rw-1
7	6	5	4	3	2	1	0
ADDMASKx							
rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1	rw-1
Modify only when UCSWRST = 1.							

**Table 24-16. UCBxADDMASK Register Description**

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	ADDMASKx	RW	3FFh	Address Mask Register. By clearing the corresponding bit of the own address, this bit is a don't care when comparing the address on the bus to the own address. Using this method, it is possible to react on more than one slave address. When all bits of ADDMASKx are set, the address mask feature is deactivated. Modify only when UCSWRST = 1.

#### 24.4.14 UCBxI2CSA Register

eUSCI\_Bx I<sup>2</sup>C Slave Address Register

**Figure 24-30. UCBxI2CSA Register**

15	14	13	12	11	10	9	8
Reserved						I2CSAx	
r-0	r0	r0	r0	r0	r0	rw-0	rw-0
7	6	5	4	3	2	1	0
I2CSAx							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

**Table 24-17. UCBxI2CSA Register Description**

Bit	Field	Type	Reset	Description
15-10	Reserved	R	0h	Reserved
9-0	I2CSAx	RW	0h	I <sup>2</sup> C slave address. The I2CSAx bits contain the slave address of the external device to be addressed by the eUSCI_B module. It is only used in master mode. The address is right justified. In 7-bit slave addressing mode, bit 6 is the MSB and bits 9-7 are ignored. In 10-bit slave addressing mode, bit 9 is the MSB.

#### 24.4.15 UCBxIE Register

eUSCI\_Bx I2C Interrupt Enable Register

**Figure 24-31. UCBxIE Register**

15		14		13		12		11		10		9		8	
Reserved		UCBIT9IE		UCTXIE3		UCRXIE3		UCTXIE2		UCRXIE2		UCTXIE1		UCRXIE1	
r0		rw-0		rw-0		rw-0		rw-0		rw-0		rw-0		rw-0	
7		6		5		4		3		2		1		0	
UCCLTOIE		UCBCNTIE		UCNACKIE		UCALIE		UCSTPIE		UCSTTIE		UCTXIE0		UCRXIE0	
rw-0		rw-0		rw-0		rw-0		rw-0		rw-0		rw-0		rw-0	

**Table 24-18. UCBxIE Register Description**

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved
14	UCBIT9IE	RW	0h	Bit position 9 interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
13	UCTXIE3	RW	0h	Transmit interrupt enable 3 0b = Interrupt disabled 1b = Interrupt enabled
12	UCRXIE3	RW	0h	Receive interrupt enable 3 0b = Interrupt disabled 1b = Interrupt enabled
11	UCTXIE2	RW	0h	Transmit interrupt enable 2 0b = Interrupt disabled 1b = Interrupt enabled
10	UCRXIE2	RW	0h	Receive interrupt enable 2 0b = Interrupt disabled 1b = Interrupt enabled
9	UCTXIE1	RW	0h	Transmit interrupt enable 1 0b = Interrupt disabled 1b = Interrupt enabled
8	UCRXIE1	RW	0h	Receive interrupt enable 1 0b = Interrupt disabled 1b = Interrupt enabled
7	UCCLTOIE	RW	0h	Clock low timeout interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
6	UCBCNTIE	RW	0h	Byte counter interrupt enable. 0b = Interrupt disabled 1b = Interrupt enabled
5	UCNACKIE	RW	0h	Not-acknowledge interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
4	UCALIE	RW	0h	Arbitration lost interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
3	UCSTPIE	RW	0h	STOP condition interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

**Table 24-18. UCBxE Register Description (continued)**

Bit	Field	Type	Reset	Description
2	UCSTTIE	RW	0h	START condition interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
1	UCTXIE0	RW	0h	Transmit interrupt enable 0 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE0	RW	0h	Receive interrupt enable 0 0b = Interrupt disabled 1b = Interrupt enabled

#### **24.4.16 UCBxIFG Register**

eUSCI\_Bx I<sup>2</sup>C Interrupt Flag Register

**Figure 24-32. UCBxIFG Register**

15	14	13	12	11	10	9	8
Reserved	UCBIT9IFG	UCTXIFG3	UCRXIFG3	UCTXIFG2	UCRXIFG2	UCTXIFG1	UCRXIFG1
r0	rw-0	rw-1	rw-0	rw-1	rw-0	rw-1	rw-0
7	6	5	4	3	2	1	0
UCCLTOIFG	UCBCNTIFG	UCNACKIFG	UCALIFG	UCSTPIFG	UCSTTIFG	UCTXIFG0	UCRXIFG0
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1	rw-0

**Table 24-19. UCBxIFG Register Description**

Bit	Field	Type	Reset	Description
15	Reserved	R	0h	Reserved
14	UCBIT9IFG	RW	0h	Bit position 9 interrupt flag 0b = No interrupt pending 1b = Interrupt pending
13	UCTXIFG3	RW	1h	eUSCI_B transmit interrupt flag 3. UCTXIFG3 is set when UCBxTXBUF is empty in slave mode, if the slave address defined in UCBxI2COA3 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
12	UCRXIFG3	RW	0h	Receive interrupt flag 3. UCRXIFG3 is set when UCBxRXBUF has received a complete byte in slave mode and if the slave address defined in UCBxI2COA3 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
11	UCTXIFG2	RW	0h	eUSCI_B transmit interrupt flag 2. UCTXIFG2 is set when UCBxTXBUF is empty in slave mode, if the slave address defined in UCBxI2COA2 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
10	UCRXIFG2	RW	0h	Receive interrupt flag 2. UCRXIFG2 is set when UCBxRXBUF has received a complete byte in slave mode and if the slave address defined in UCBxI2COA2 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
9	UCTXIFG1	RW	1h	eUSCI_B transmit interrupt flag 1. UCTXIFG1 is set when UCBxTXBUF is empty in slave mode, if the slave address defined in UCBxI2COA1 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
8	UCRXIFG1	RW	0h	Receive interrupt flag 1. UCRXIFG1 is set when UCBxRXBUF has received a complete byte in slave mode and if the slave address defined in UCBxI2COA1 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
7	UCCLTOIFG	RW	0h	Clock low timeout interrupt flag 0b = No interrupt pending 1b = Interrupt pending
6	UCBCNTIFG	RW	0h	Byte counter interrupt flag. When using this interrupt the user needs to ensure enough processing bandwidth (see the Byte Counter Interrupt section). 0b = No interrupt pending 1b = Interrupt pending

**Table 24-19. UCBxIFG Register Description (continued)**

Bit	Field	Type	Reset	Description
5	UCNACKIFG	RW	0h	Not-acknowledge received interrupt flag. This flag only is updated when operating in master mode. 0b = No interrupt pending 1b = Interrupt pending
4	UCALIFG	RW	0h	Arbitration lost interrupt flag 0b = No interrupt pending 1b = Interrupt pending
3	UCSTPIFG	RW	0h	STOP condition interrupt flag 0b = No interrupt pending 1b = Interrupt pending
2	UCSTTIFG	RW	0h	START condition interrupt flag 0b = No interrupt pending 1b = Interrupt pending
1	UCTXIFG0	RW	0h	eUSCI_B transmit interrupt flag 0. UCTXIFG0 is set when UCBxTXBUF is empty in master mode or in slave mode, if the slave address defined in UCBxI2COA0 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending
0	UCRXIFG0	RW	0h	eUSCI_B receive interrupt flag 0. UCRXIFG0 is set when UCBxRXBUF has received a complete character in master mode or in slave mode, if the slave address defined in UCBxI2COA0 was on the bus in the same frame. 0b = No interrupt pending 1b = Interrupt pending

#### **24.4.17 UCBxIV Register**

eUSCI\_Bx Interrupt Vector Register

**Figure 24-33. UCBxIV Register**

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
UCIVx							
r0	r0	r0	r0	r-0	r-0	r-0	r0

**Table 24-20. UCBxIV Register Description**

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI_B interrupt vector value. It generates an value that can be used as address offset for fast interrupt service routine handling. Writing to this register clears all pending interrupt flags. 00h = No interrupt pending 02h = Interrupt Source: Arbitration lost; Interrupt Flag: UCALIFG; Interrupt Priority: Highest 04h = Interrupt Source: Not acknowledgment; Interrupt Flag: UCNACKIFG 06h = Interrupt Source: Start condition received; Interrupt Flag: UCSTTIFG 08h = Interrupt Source: Stop condition received; Interrupt Flag: UCSTPIFG 0Ah = Interrupt Source: Slave 3 Data received; Interrupt Flag: UCRXIFG3 0Ch = Interrupt Source: Slave 3 Transmit buffer empty; Interrupt Flag: UCTXIFG3 0Eh = Interrupt Source: Slave 2 Data received; Interrupt Flag: UCRXIFG2 10h = Interrupt Source: Slave 2 Transmit buffer empty; Interrupt Flag: UCTXIFG2 12h = Interrupt Source: Slave 1 Data received; Interrupt Flag: UCRXIFG1 14h = Interrupt Source: Slave 1 Transmit buffer empty; Interrupt Flag: UCTXIFG1 16h = Interrupt Source: Data received; Interrupt Flag: UCRXIFG0 18h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG0 1Ah = Interrupt Source: Byte counter zero; Interrupt Flag: UCBCNTIFG 1Ch = Interrupt Source: Clock low timeout; Interrupt Flag: UCCLTOIFG 1Eh = Interrupt Source: 9th bit position; Interrupt Flag: UCBIT9IFG; Priority: Lowest

## Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

### Changes from July 26, 2016 to December 12, 2016

### Page

• Added DISOOFP (bit 9) and DISFPCA (bit 8) bits to <a href="#">Section 2.4.6.2, ACTLR Register</a> .....	162
• Corrected typo in SVSMH bit name in <a href="#">Section 3.3.8, RSTCTL_PSSRESET_STAT Register</a> .....	256
• Added that encrypted payload must be written to Bank 1 of device memory in <a href="#">Section 4.8.5.3.2, Encrypted Update: JTAG and SWD Locked Device</a> .....	267
• Added that payload must be written to Bank 1 of device memory in <a href="#">Section 4.8.5.4.1, Unencrypted Update: IP Protected Device</a> .....	268
• Added that payload must be written to Bank 1 of device memory in <a href="#">Section 4.8.5.4.2, Encrypted Update: IP Protected Device</a> .....	268
• Added that SEC_ZONE <sub>x</sub> _START_ADDR should be in Bank 0.....	272
• Added that SEC_ZONE <sub>x</sub> _ENCPAYLOADADDR should be in Bank 1.....	274
• Added "should be in Bank 0 of the main flash of the device" to the start address list item in <a href="#">Section 4.8.6.2.2, Boot Override Commands and Acknowledgments</a> .....	277
• Added "in steps of 4KB" to the zone length list item in <a href="#">Section 4.8.6.2.2, Boot Override Commands and Acknowledgments</a> .....	277
• Added <a href="#">Section 5.2.3.1, Using HFXT Oscillator After Wakeup From Low-Power Modes</a> .....	308

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products	Applications
Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>
	<b>TI E2E Community</b>
	<a href="http://e2e.ti.com">e2e.ti.com</a>